UFC-UNIVERSIDADE FEDERAL DO CEARÁ
CT-CENTRO DE TECNOLOGIA
DEE-DEPARTAMENTO DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JOSÉ ARILO RIBEIRO LANDIM JÚNIOR

DISPOSITIVO DE COMUNICAÇÃO USB PARA SUPERVISÃO E CONTROLE DE SISTEMAS VIA COMPUTADOR

FORTALEZA
2014

JOSÉ ARILO RIBEIRO LANDIM JÚNIOR

DISPOSITIVO DE COMUNICAÇÃO USB PARA SUPERVISÃO E CONTROLE DE SISTEMAS VIA COMPUTADOR

Trabalho de conclusão de curso em Engenharia Elétrica da Universidade Federal do Ceará, como pré-requisito para a obtenção do Título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Paulo Peixoto Praça.

FORTALEZA

2014

Página reservada para ficha catalográfica que deve ser confeccionada após apresentação e alterações sugeridas pela banca examinadora.

Para solicitar a ficha catalográfica de seu trabalho, acesse o site: www.biblioteca.ufc.br, clique no banner Catalogação na Publicação (Solicitação de ficha catalográfica)

JOSÉ ARILO RIBEIRO LANDIM JÚNIOR

DISPOSITIVO DE COMUNICAÇÃO USB PARA SUPERVISÃO E CONTROLE DE SISTEMAS VIA COMPUTADOR: DSC – DISPOSITIVO DE SUPERVISÃO E CONTROLE

Trabalho de conclusão de curso em Engenharia Elétrica da Universidade Federal do Ceará, como pré-requisito para a obtenção do Título de Bacharel em Engenharia Elétrica.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Peixoto Praça (Orientador)
Universidade Federal do Ceará (UFC)

_____

Msc. Davi Rabelo Joca
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Tobias Rafael Fernandes Neto
Universidade Federal do Ceará (UFC)

A Deus.

Aos meus pais, Maria Orlandina e José Arilo.

AGRADECIMENTO

Sobre tudo preciso agradecer a Deus e a minha família, bases fundamentais para que eu tenha lutado para chegar até aqui, e me tornado quem sou e quem ainda posso me tornar. Como diz a música, “Cada um sabe a dor e a delícia de ser o que é....”, e vocês nos momentos de maior dor me deram forças para continuar sempre seguindo em frente. Amo todos vocês!

Ao Prof. Dr. Paulo Peixoto Praça, pela paciência e boas sugestões para que este trabalho tenha sido completado com sucesso. Aos professores participantes da banca examinadora Davi Rabelo Joca e Tobias Rafael Fernandes Neto pelo tempo gasto analisando o trabalho, pela atenção e sugestões, que servirão para meu crescimento como profissional.

A todas as amizades aqui construídas, acredito que sem elas o caminho teria sido mais difícil. Queria agradecer em especial a alguns amigos, que comigo sofreram com os trabalhos e noites estudando. Ao Túlio, que ajudou até mesmo longe de Fortaleza, Samuel, Lucas Dantas, Jefferson, mais conhecido como chorão, Igão, Obed, Emanuel, Rannier, Dimas, Jordão e Larissa. Enfim, todos são parte fundamental da conclusão de mais uma etapa de minha vida. Espero sinceramente que todas as amizades aqui construídas possam ser levadas pro resto da vida.

Não poderia esquecer de meus amigos de infância que mesmo longe por algum tempo, nunca deixaram de me apoiar e torcer por mim. Dedico a vocês também mais essa vitória em minha vida, Ana Régia, Suzy, Rômulo, Raissa, Evelise e Gardênio, principalmente.

“Por aqui, contudo, não olhamos para trás por muito tempo. Seguimos em frente, abrindo novas portas e fazendo coisas novas...e curiosidade nos conduz a novos caminhos. ”

(Walt Disney)

RESUMO

Trata-se de um dispositivo de comunicação via USB, tendo como objetivo criar um link de comunicação entre um microcontrolador e o computador. Para estudo do funcionamento do dispositivo, foi proposto a criação de um supervisório de um sistema portuário, que por meio de uma sala de controle, pode-se comandar e verificar válvulas, iluminação e nível da caixa d'água. O objetivo é desenvolver uma solução para este sistema, constituído por um hardware e um software, no qual o hardware tratará de receber e enviar os dados e o software interpretará todos as variáveis do sistema, facilitando a visualização e controle para o operador do sistema. A escolha da comunicação usb entre o software e hardware, deve-se ao fato da grande maioria dos computadores nos dias atuais possuírem apenas portas para conexão usb e não mais paralela.

Palavras-chave: USB, monitoramento e controle.

ABSTRACT

It’s a communication device for USB, having how objective to create one link of communication between microcontroller and the computer. To study the operation of the device, went proposed the creation of a supervisory of one harbor system, which through a control room can control and check valves, lighting and level of the water tanks. The objective is to develop one solution for this system, consisting of hardware and software, where the hardware will receive and send data and the software interprets all the variables of the system, facilitating the visualization and control of the system operator. The choice of USB communication between software and hardware, should of the fact that most computers nowadays only have USB ports for connection and no more parallel.

Keywords: USB, monitoring and control.

LISTA DE ILUSTRAÇÕES

Figura 1 – Fluxo de dados entre os vários periféricos e o host. 15
Figura 2 – Sistema USB, composto de Hubs e dispositivos USB. 16
Figura 3 – Níveis de hardware e software necessários para comunicação USB..... 17
Figura 4 – Sequência de eventos desde a conexão física dos dispositivos até a inicialização dos drivers. 18
Figura 5 – Principais tipos de conectores USB. 19
Figura 6 – Cabo USB. 20
Figura 7 – Exemplo de codificação NRZI. 20
Figura 8 – Cabo USB, com detalhe para o par trançado D+ e D- 20
Figura 9 – Estados lógicos definidos para comunicação USB. 21
Figura 10 – Máquina de estado para transmissão de bits. 21
Figura 11 – Resumo de eventos para envio de um pacote de dados. 22
Figura 12 – Comunicação remota entre a unidade principal e as unidades secundárias..... 26
Figura 13 – Dispositivo utilizando classe CDC de comunicação USB. 28
Figura 14 – Chipset FT 232. 29
Figura 15 – Mapa ilustrativo do estudo de caso. 30
Figura 16 – Representação de cabos de ethernet. 31
Figura 17 - Rádio XTend-PKG. 32
Figura 18 – Diagrama de blocos do funcionamento do DSC. 33
Figura 19 – Visão geral da unidade principal e suas funções. 34
Figura 20 – Esquema de acionamento de um relé. 34
Figura 21 – Microcontrolador utilizado para estabelecer a comunicação com o supervisório. 35
Figura 22 – Oscilador do DSC. 36
Figura 23 – Sinaleira de estado do DSC. 36
Figura 24 – Sensores da caixa D'Água inferior e superior. 37
Figura 25 – Representação de relés para acionamento das válvulas. 38
Figura 26 – Comando para emergências. 39
Figura 27 – Esquema utilizado para ilustrar as torres de iluminação. 40
Figura 28 – Componente de chaveamento IGBT. 40

Figura 29 – Diagrama de blocos da programação do supervisório. 41
Figura 30 – Tela de Bloqueio. 42
Figura 31 – Tela principal. 43
Figura 32 – Tela sobre. 44
Figura 33 – Tela do manual. 45
Figura 34 – Tela de abastecimento. 47
Figura 35 – Tela de recalque. 48
Figura 36 – Tela de iluminação. 49
Figura 37 – Simulação da comunicação entre o supervisório e o DSC. 50

LISTA DE ABREVIATURAS E SIGLAS

CDC	Connected Device Configuration
CLP	Controlador Lógico Programável
DLL	Dynamic-link library
DSC	Dispositivo de Supervisão e Controle
EHCI	Enhanced Host Controller Interface
HID	Human Interface Device
NRZI	Non Return to Zero Inverted
OHCI	Open Host Controller Interface
PC	Personal Computer
PID	Product ID
RF	Rádio Frequência
SCADA	Supervisory Control And Data Acquisition
UHCI	Universal Host Controller Interface
USB	Universal Serial Bus
VID	Vendor ID

LISTA DE SÍMBOLOS

% Porcentagem

© Copyright

® Marca Registrada

SUMÁRIO

1. INTRODUÇÃO..... 13

2. COMUNICAÇÃO VIA USB..... 14

2.1. Barramento USB..... 15

2.2. Controlador Host..... 16

2.3. Tipos de conectores USB..... 19

2.4. Estruturas elétrica e Sinais do cabo USB..... 19

2.5. Tipos de Pacotes..... 22

2.6. Tipos de Transação de dados..... 23

2.7. Modos de operação do dispositivo - Bus- Powered e Self-Powered..... 23

2.8. Endpoints e Pipes..... 24

2.9. Descritores..... 24

2.10. AVANÇO TECNOLÓGICO NA COMUNICAÇÃO USB E EM SISTEMAS DE SUPERVISÃO..... 25

2.10.1. *Comunicação USB utilizando classe CDC*..... 27

2.10.2. *Comunicação USB utilizando FT 232*..... 28

3. ESTUDOS DE CASO – SUPERVISÃO SISTEMA PORTUÁRIO..... 29

3.1. Detalhamentos do Caso base..... 29

3.2. Detalhamentos do Projeto..... 32

4. DESCRIÇÕES DO DISPOSITIVO DE SUPERVISÃO E CONTROLE..... 33

4.1. Unidade Principal do DSC– Protocolo de Comunicação..... 35

4.2. Unidades Secundárias do DSC – Sensores e relés..... 36

4.2.1. *Sensores para Caixa D'Água*..... 37

4.2.2. *Relés para acionamento das válvulas*..... 38

4.2.3. *Relés para acionamento da Iluminação*..... 39

5. SUPERVISÓRIO DO DSC..... 40

5.1. Tela de Bloqueio..... 42

5.2. Tela Principal..... 43
5.3. Tela Sobre..... 44
5.4. Tela do Manual..... 45
5.5. Tela Modo Abastecimento..... 46
5.6. Tela Modo Recalque..... 48
5.7. Tela Modo Iluminação..... 49
6. RESULTADOS DE SIMULAÇÃO..... 50
7. CONCLUSÕES..... 52
REFERÊNCIAS..... 53
ANEXO A – INSTRUÇÕES BIBLIOTECA MCHID.DLL..... 55
ANEXO B – INSTRUÇÕES DAS BIBLIOTECAS USB PIC..... 59

1. Introdução

Nos últimos anos, é notável o grande avanço da tecnologia. Com a utilização de dispositivos eletrônicos para solucionar os mais diversos problemas. Com isso, a área de automação experimenta um grande avanço, devido a quantidade de novas tecnologias sendo desenvolvidas. Contudo, a simples utilização de dispositivos para automação, como Controladores Lógicos Programáveis (CLPs), não permite a visualização, de forma geral, do processo que estar sendo executado, para isso tornou-se necessário a criação de interface homem-máquina, mais conhecido como IHM, que permite a visualização, de forma simples e clara, das atividades executadas.

Contudo, ainda é difícil o desenvolvimento de novos dispositivos e softwares. Devido ao fato da existência de poucos componentes eletrônicos que permitam a criação de dispositivos que possam se comunicar com o computador de forma simples, e de uma plataforma de desenvolvimento de softwares que facilitem a criação destes.

Desta forma, este trabalho tem como objetivo apresentar um método que possa facilitar o desenvolvimento de um dispositivo que possa ser utilizado para automações, tendo em vista a substituição de um CLP, como também o desenvolvimento de um software para computadores, que poderá estabelecer comunicação com o dispositivo e possibilitando a observação do processo e de controlá-lo.

Para a solução do problema do *hardware* são utilizados microcontroladores que possuem interface USB (Universal Serial Bus), haja visto que a grande maioria dos computadores modernos possuem várias portas destas que possam permitir a comunicação. Já para solucionar o problema do *software*, utiliza-se o Microsoft Visual Basic, que com a ajuda de uma DLL desenvolvida para auxiliar a comunicação USB, permite a comunicação mais fácil com o dispositivo.

Após a análise de todos esses aspectos pode-se verificar que este trabalho, não está limitado ao estudo de caso aqui abordado e podendo ter uma abrangência maior, podendo ser utilizado da escala residencial a industrial, uma vez que a tecnologia aqui desenvolvida é de baixo custo.

Este trabalho foi dividido em quatro partes principais. A primeira parte terá como base abordar todas a informações utilizadas para desenvolver o hardware, que irá estabelecer comunicação com o computador via USB. A segunda parte trará as informações para o desenvolvimento do *software* do computador, conhecido como IHM ou supervisório. A

terceira parte terá como objetivo mostrar um estudo de caso ao qual o dispositivo aqui desenvolvido foi aplicado. A parte final irá mostrar os resultados obtidos através de simulação, devido ao desenvolvimento do primeiro protótipo.

Vale ressaltar que o trabalho desenvolvido é apenas um protótipo, sendo assim necessário adequá-lo às normas reguladoras para que possa ser produzido em escala industrial.

2. Comunicação via USB

O USB surgiu em 1995 por conta de uma parceria entre várias companhias de alta tecnologia, entre elas estão Compaq, Hewlett-Packard, Intel, Microsoft e etc. Uma das primeiras versões foi a 1.0 com velocidade de 1,5Mbs, logo em seguida foi concebida a 1.1 com velocidade que vai de 1,5Mbps a 12 Mbps. No final de 2000 foi lançada a versão 2.0, compatível com as versões anteriores, mas alguns aperfeiçoamentos que vão desde a topologia, à velocidade de trafego de dados, chegando ao extremo de 480Mbps.

As primeiras versões do USB utilizavam os Controladores Host UHCI (Universal Host Controller Interface), e OHCI (Open Host Controller Interface). O USB 2.0 utiliza o Controlador HOST EHCI (Enhanced Host Controller Interface) (MESSIAS, 2014).

A ideia do padrão de interface USB foi criada e amadurecida para suprir algumas necessidades e oferecer características confortáveis de operação, não presentes nos padrões anteriores, como (ZELENOVSKY; MENDONÇA, 2006):

- Permitir a conexão de vários periféricos sem a necessidade de abrir o gabinete do computador.
- Permitir a adição de dispositivo “on the fly”, ou seja, com o sistema operacional já rodando aplicativos.
- Oferecer um tipo único de conector, simples e barato, para encaixe.
- Implementar um controle interno de consumo, onde o periférico USB automaticamente se desconecte, não consumindo potência, quando estiver ocioso.
- Especificar um protocolo de comunicação com detecção e supressão automática de erros de transmissão.

2.1. Barramento USB

O barramento USB permite a conexão de até no máximo 127 dispositivos em uma única porta. Para isso, utiliza-se *Hubs*, que conectados em cascata tornam isso possível. Os quais, em sua grande maioria, dispõem de 4 e 8 portas. Os *Hubs* são componentes importantes na topologia de uma rede USB, pois eles fornecem novos canais físicos para que se possa inserir novos dispositivos à uma única porta de entrada. Na figura 1, é possível ver como se dá o fluxo de dados entre os vários periféricos e o *host* (ZELENOVSKY; MENDONÇA, 2006).

Figura 1 – Fluxo de dados entre os vários periféricos e o *host*.

Fonte: Dj Techtools.

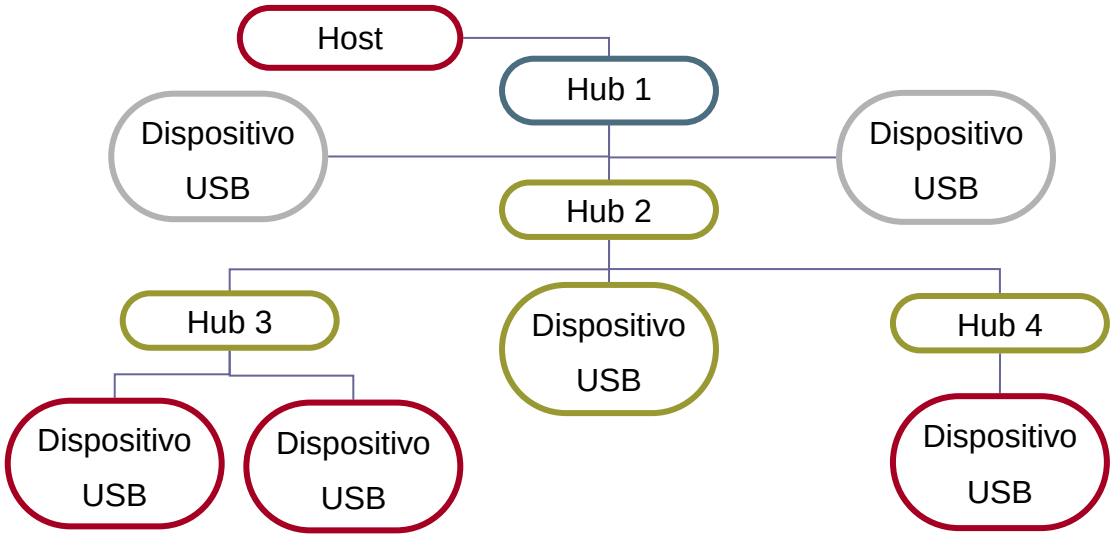
Os *Hubs*, na grande maioria, são ligados à rede elétrica para alimentar seus circuitos e ao mesmo tempo fornecer corrente suficiente para alimentar os dispositivos conectados às suas portas. Alguns *Hubs* não são conectados à rede, sendo seus circuitos alimentados a partir da porta USB na qual está conectado. Contudo, estes modelos de hubs não são aconselháveis, se no barramento for conectado dispositivos que alimentam-se através do mesmo.

Os *Hubs* sem fonte externa de alimentação chegam a ter 4 portas *downstream*, fluxo de corrente sai da porta USB, fornecendo cada uma 100mA, já os modelos com fontes de alimentação podem fornecer por porta, até 500mA. Se um dispositivo necessita de uma

corrente superior à que o Hub pode fornecer, este permanecerá conectado fisicamente, mas não haverá comunicação (MESSIAS, 2014).

Em um sistema USB existe apenas um *Host*, os demais componentes são os *Hub* e os dispositivos USB, como verifica-se na figura 2.

Figura 2 – Sistema USB, composto de *Hubs* e dispositivos USB.



Fonte: Elaborado pelo Autor.

2.2. Controlador *Host*

O computador comunica-se com os dispositivos através do seu controlador *Host*, sendo este encontrado na própria estrutura do computador.

É de responsabilidade do *Host*:

- Detectar a inclusão e remoção de dispositivos;
- Gerenciar o fluxo de controle de dados entre os dispositivos conectados;
- Fornecer alimentação aos dispositivos conectados;
- Monitorar os sinais do *bus* USB.

Como já foi mencionado previamente o padrão USB foi desenvolvido por um consórcio de empresas de tecnologia, que após várias discussões sobre controlador, terminou criando dois grupos, o UHCI e OHCI.

O controlador UHCI era defendido pelas empresas participantes, que concordavam que uma parte do funcionamento do protocolo deveria ser processado no software, tornando o *hardware* do controlador mais simples. E o controlador OHCI era defendido por aqueles que concordavam que maior parte do processamento deveria ser executado pelo controlador, simplificando o *driver*.

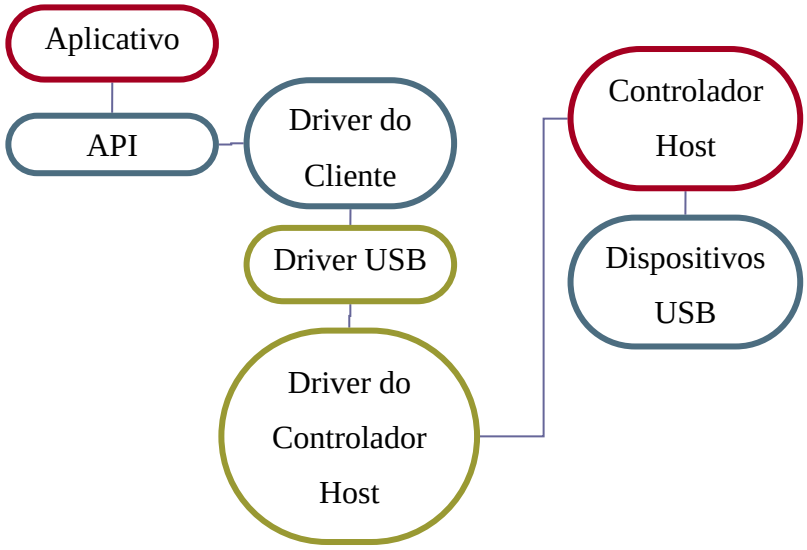
Por causa deste desacordo entre as empresas, foi gerado uma incompatibilidade no padrão USB. Para solucionar este problema foi criado o controlador EHCI, que surgiu para integrar o que há de melhor de cada um dos modelos já existentes.

Estes controladores são compostos por um *software*, cujo driver é fornecido pelo fabricante, e um *hardware* que processa os sinais do barramento USB.

Os controladores UHCI e OHCI foram muito utilizados na versão 1.1, já o EHCI foi utilizado na versão 2.0. O barramento 2.0 é compatível com os controladores UHCI e OHCI. Desta forma, o EHCI tornou-se um padrão USB universal (MESSIAS, 2014).

A arquitetura USB é bem simples, apesar de ser um sistema constituído por vários níveis de *hardware* e *software*, conforme mostrado na figura 3.

Figura 3 – Níveis de hardware e software necessários para comunicação USB.



Fonte: Elaborado pelo autor.

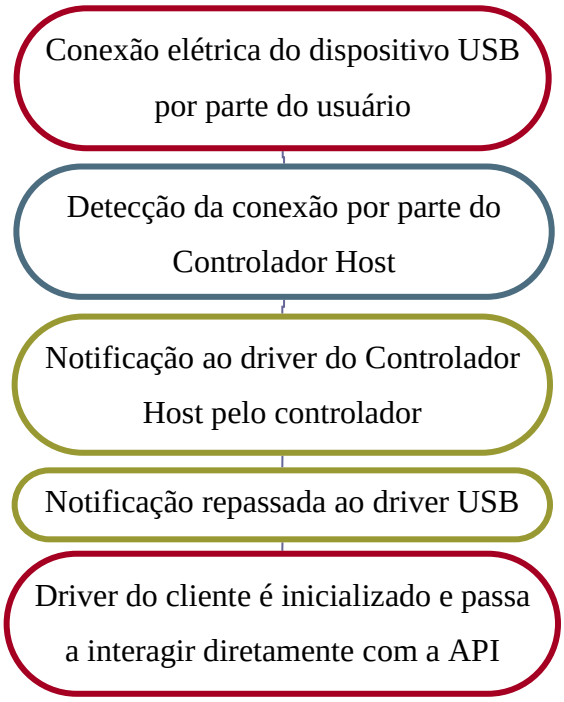
Fazendo uma rápida análise do diagrama da figura 3, tem-se em primeiro lugar, do lado do *software*, o aplicativo, onde será tratado todos os dados recebido do dispositivo. Na sequência encontra-se o API (Application Programming Interface), que é uma coleção de rotinas com chamadas padronizadas, que os aplicativos utilizam para lançar mão dos recursos

do computador e do sistema operacional. Em terceiro lugar encontra-se o *driver* do cliente, que irá depender do tipo de periférico, sendo este a interface entre as rotinas do API e as rotinas USB. Em quarto lugar está o *driver* USB, que é a camada que torna o acesso ao controlador *host* mais amigável. E por último, do lado do *software*, encontra-se o *driver* do controlador *host*, uma interface de software que realiza os acessos de I/O propriamente ditos.

Do lado do *hardware*, está o controlador *host*, que é um circuito que processa eletronicamente e deixa disponíveis os dados trafegados via USB. E por fim o dispositivo USB, que recebe ou envia dados do aplicativo (ZELENOVSKY; MENDONÇA, 2006).

E ainda de forma mais simples, a comunicação USB pode ser entendida pelo diagrama da figura 4 (ZELENOVSKY; MENDONÇA, 2006).

Figura 4 – Sequência de eventos desde a conexão física dos dispositivos até a inicialização dos drivers.


Conexão elétrica do dispositivo USB por parte do usuário
Detecção da conexão por parte do Controlador Host
Notificação ao driver do Controlador Host pelo controlador
Notificação repassada ao driver USB
Driver do cliente é inicializado e passa a interagir diretamente com a API

Fonte: Elaborado pelo autor.

2.3. Tipos de conectores USB

Existe dois grupos de conectores USB, sendo estes tipos A e B.

Os conectores tipo A fêmea são utilizados no PC ou em portas *Downstream*, fornecem corrente a outros dispositivos, de *Hubs* e o macho em cabos USB, onde deve ser conectado ao PC ou portas *Downstream* de *Hubs*. Os conectores tipo B fêmea, são mais encontrados em dispositivos de função e o macho é encontrado em um dos extremos do cabo, onde deve ser conectado a um dispositivo. Na figura 5, pode-se ver os quatro principais tipos de conectores (MESSIAS, 2014).

Figura 5 – Principais tipos de conectores USB.

Fonte: Elaborado pelo autor.

Os conectores USB foram projetados mecanicamente de forma que não seja permitido a conexão errada por parte do usuário.

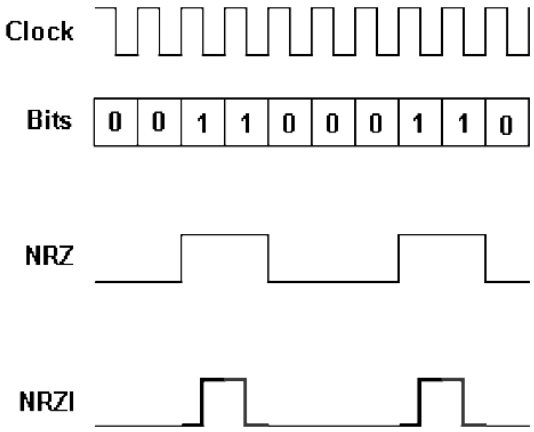
2.4. Estruturas elétrica e Sinais do cabo USB

O cabo USB, mostrado na figura 6, é composto por quatro fios e uma malha para eliminar ruídos. Dois desses fios são utilizados para transportar energia para alimentar o dispositivo. Os outros dois fios, D+ e D-, são usados pelo sistema USB para transferência de dados entre o PC, *hub* e dispositivos (MESSIAS, 2014). Todos os dados trafegam utilizando a codificação NRZI (Non Return to Zero Inverted), estando ilustrado este tipo de codificação na figura 7.

Figura 6 – Cabo USB.

Fonte: Elaborada pelo Autor.

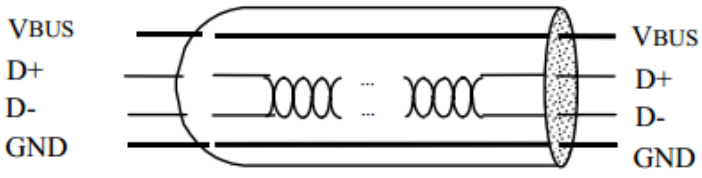
Figura 7 – Exemplo de codificação NRZI.



Fonte: Curso Online USB.

É utilizado também um padrão de cores para o cabo. O fio de cor vermelha é chamado de Vbus, sendo este o fio positivo de fornecimento de energia. O fio preto é o GND, sendo este o fio negativo de fornecimento de energia. O D+ é da cor verde e o D- é da cor branca. Vale ressaltar que os cabos de dados, D+ e D-, são um par trançado, desse modo reduzem o ruído e interferência existente entre eles, como mostrado na figura 8 (COMPAQ; INTEL; MICROSOFT; NEC, 2014).

Figura 8 – Cabo USB, com detalhe para o par trançado D+ e D-.



Fonte: Universal Serial Bus Specification.

A Comunicação entre o *Host* e o periférico tem definido três estados lógicos para comunicação a 12Mb/s, como mostrado na figura 9.

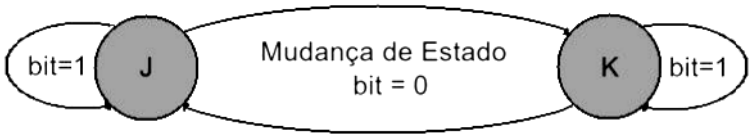
Figura 9 – Estados lógicos definidos para comunicação USB.

Estado	D+	D-
J	Alto	Baixo
K	Baixo	Alto
SE0	Baixo	Baixo

Fonte: Elaborado pelo autor.

Um barramento que está ocioso se encontra no estado J, enquanto um barramento que está inativo se encontra no estado SE0. A regra para transforma J's e K's em 1's e 0's é simples, apesar de não ser imediata. Utilizando a codificação NRZI, para transmitir um *bit* zero é necessário que se invertam os estados nas linhas D+ e D-, por outro lado, para transmitir o *bit* um os estados devem ser mantidos. Essa mudança pode ser melhor entendida através da figura 10.

Figura 10 – Máquina de estado para transmissão de bits.



Fonte: Elaborado pelo autor.

Como por exemplo, na sequência de sincronismo, KJKJKJKK = 00000001.

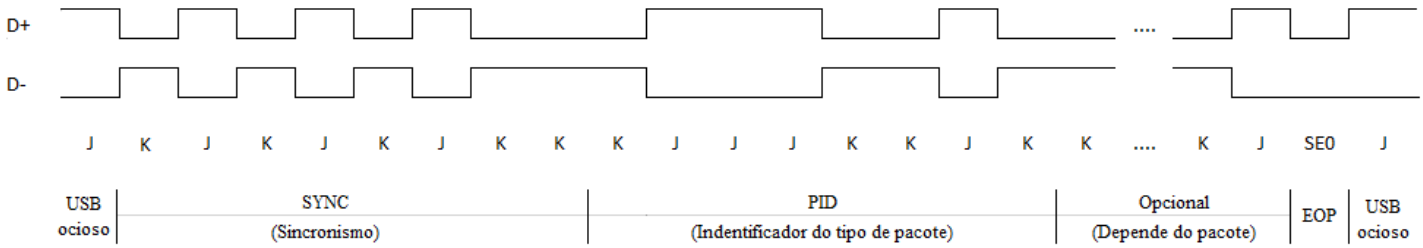
Para evitar que o barramento permaneça no estado J por muito tempo, julgou-se conveniente criar o *bit* redundante, que deve ser desprezado na recepção. A cada seis *bits* 1's é transmitido um *bit* zero de forma redundante, também conhecido como *bit stuff*.

Para envio de um pacote de dados através da comunicação USB, é necessário no mínimo três pacotes principais: sincronismo (SYNC), identificador (PID) e o pacote de dados. Cada pacote se inicia com uma sequência de sincronização de 8 *bits*, que corresponde à sequência de estados KJKJKJKK, como já mencionado anteriormente. Antes de qualquer transmissão, o cabo e a interface estão no estado J, ou ocioso.

Em seguida é feita a transmissão do PID, também de 8 *bits*. Os 4 primeiros *bits* definem o tipo do pacote. E os 4 últimos, são construídos a partir dos quatro estados dos *bits* anteriores, através do complemento dos *bits* que os estados representam. Logo na sequência é enviado o pacote com o dado e encerrado com transmissão do pacote EOP (End of Packet), que é composto por dois estados SE0. Se o host mantiver três estados SE0 consecutivos, então o periférico será considerado desconectado fisicamente.

Se houver mais dados a serem transmitidos, após o pacote EOP, é reiniciado a sequência de envio de pacotes (ZELENOVSKY; MENDONÇA, 2006). Na figura 11, pode-se visualizar o resumo de eventos para envio de um único dado.

Figura 11 – Resumo de eventos para envio de um pacote de dados.


D+
D-
J K J K J K J K K K J J J K K J K K .... K J SE0 J
USB ocioso
SYNC (Sincronismo)
PID (Indentificador do tipo de pacote)
Opcional (Depende do pacote)
EOP
USB ocioso

Fonte: Elaborado pelo autor.

2.5. Tipos de Pacotes

A versão 2.0 do USB prevê 9 identificadores de pacotes para todas as velocidades, que estão agrupados em 3 categorias: *Token*, *Data* e *Handshake* (MESSIAS, 2014).

- Tipo *Token* (*In*, *Out*, *Setup*): Esses pacotes tem a finalidade de selecionar qual será o dispositivo alvo da próxima transação. O *Token* In informa que o próximo pacote de dados será para leitura. O *Token Out* indica que a transação será para escrita. Já o *Token Setup* é utilizado para iniciar transferências e controle, sempre endereçado ao endpoint zero.

- Tipo *Data* (*DATA0, DATA1*): Esses pacotes são os responsáveis por carregar a informação propriamente dita, contudo, deve ser respeitado o fluxo de dados definido pelo último Token.

- Tipo *Handshake* (ACK, NAK, STALL): Os pacotes desse tipo são os mais simples, constituído apenas pelos campos obrigatórios de um pacote de transmissão USB, SYNC, PID e EOP. O fluxo de transmissão é sempre o inverso do fluxo definido pelo último *Token*, pois os pacotes *Handshake* tem a finalidade de avisar a quem enviou os dados sobre a situação do

recebimento. O pacote ACK informa que o último pacote foi recebido corretamente. O pacote NAK informa que o dispositivo está ocupado ou inapto para realizar comunicação no momento. Um pacote *STALL* ocorre quando foi identificado pelo receptor erros de comunicação.

- SOF (*Start of Frame*): Esse pacote é transmitido pela raiz a uma taxa de 1000 pacotes/segundo, não possuindo grandes implicações práticas senão para manter atividade no barramento.

2.6. Tipos de Transação de dados

O USB possui três principais modos de operação: *Interrupt*, *Bulk* e *Isochronous*.

O Interrupt é um modo de alta prioridade, no qual é reservado parte da banda disponível para dispositivos de entrada, como teclados e mouses, mantendo assim, sempre um canal descongestionado.

O modo *Isochronous* é destinado a transmitir uma quantidade relativamente pequena de dados que necessitam de certa prioridade. Aconselhável utilizá-lo em transações de informação para monitores de vídeos ou periféricos de áudio.

No modo *Bulk* são trafegados grandes pacotes de dados e com baixa prioridade. Um exemplo de utilização são os discos rígidos externo. A banda disponível para esse modo é a banda restante dos canais dos outros dois modos, pois os anteriores têm preferência (ZELENOVSKY; MENDONÇA, 2006).

2.7. Modos de operação do dispositivo - *Bus- Powered* e *Self-Powered*

Um dispositivo USB pode trabalhar de dois modos, *Bus-Powered* ou *Self-Powered*. No modo *Bus-Powered* o dispositivo USB não possui uma fonte de alimentação própria, sendo necessário estar conectado ao barramento de força da porta USB e estando limitado a uma corrente máxima de consumo de 100mA. Sendo assim, o modo *Self-Powered* funciona de forma contrária, não havendo limite de corrente desde que a fonte externa suporte tal demanda. Para determinação do modo de operação, é necessário que o desenvolvedor configure o circuito eletricamente de tal maneira que o dispositivo trabalhe no modo desejado.

À critério de projeto, foi escolhido o modo *Self-Powered*, pois assim garante-se uma maior segurança para a porta USB do computador e dá maior liberdade para desenvolvimento do dispositivo, não havendo limitação de corrente (MESSIAS, 2014).

2.8. *Endpoints* e *Pipes*

Endpoint é a área da memória reservada no dispositivo para armazenar os dados que trafegam em uma pipe. Contudo, um dispositivo USB pode ter no máximo 16 *endpoints*, versão 2.0, dos quais existe um reservado para comunicação com o computador, afim de enviar suas informações de identificação do dispositivo, tais como número de série, fabricante, classe, nome do produto e outros. Vale ressaltar que somente após a aquisição de todas as informações de identificação do dispositivo pelo computador, ocorre o estabelecimento real de uma comunicação USB.

Pipe é uma associação entre um *Endpoint* do dispositivo e um software no computador. Quando o dispositivo é reconhecido pelo computador, esta via de comunicação é criada, contudo, vale ressaltar que *pipe* não é uma ligação física, existindo apenas a nível de comunicação virtual (MESSIAS, 2014).

2.9. Descritores

Todos os dispositivos USB possuem descritores que informam ao computador todas as informações necessárias para identificá-lo e estabelecer uma conexão. Sendo importante ressaltar duas características o VID (ID do Vendedor) e o PID (ID do Produto), que são duas informações muito utilizadas no desenvolvimento do software que irá se comunicar com o dispositivo neste projeto.

Os descritores utilizados nesse projeto são de modelos que foram obtidos junto ao compilador utilizado, alterando-se apenas o VID e PID, já que tais informações são necessárias para o desenvolvimento do software do computador (MESSIAS, 2014).

2.10. Avanço tecnológico na comunicação USB e em Sistemas de Supervisão

Com os avanços tecnológicos, a comunicação serial e paralelas começaram a entrar em declínio, pois os computadores mais modernos já não possuíam mais tais entradas, possuindo somente portas USB, as quais podem ser ampliadas com utilização *Hubs*. Contudo, em comparação com as tecnologias mais antigas de comunicação, a utilização da tecnologia USB é bem mais complexa, do ponto de vista do desenvolvimento de *hardware* e *software*, sendo seu uso pouco propagado até o momento entre os meios de desenvolvimento científicos a nível de universidades.

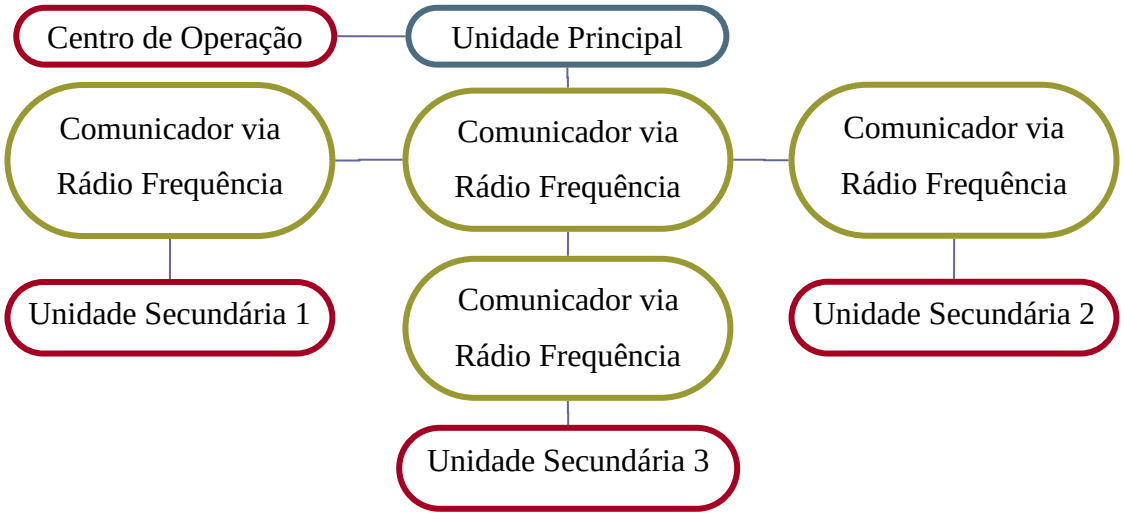
Apesar de a comunicação USB ser amplamente utilizada, nos sistemas de supervisão esta tecnologia é pouco utilizada, sendo mais utilizada a comunicação via *ethernet*.

O desenvolvimento de sistemas de supervisão ainda é uma área em expansão, pois é necessário garantir maior confiabilidade na transmissão dos dados, devido as interferências eletromagnéticas, mesmo em cabos protegidos (CASILLO,2011).

Entretanto, apesar das dificuldades para desenvolvimento de sistemas de supervisão, os mais modernos existentes possuem telemetria, que conecta o supervisório aos equipamentos, haja visto que estes estão separados por grandes distâncias. Sendo assim, é possível adquirir dados e controlar processos mais facilmente.

Um sistema SCADA (Supervisory Control And Data Acquisition), é um sistema constituído por unidades terminais remotas, sendo estas na maioria das vezes CLPs, coletando dados de campo e transmitindo a uma estação mestre via um sistema de comunicação. A estação principal recebe todos os dados e apresenta, possibilitando assim o operador controlar tudo remotamente. Através da figura 12, pode-se verificar na forma de diagrama a comunicação entre a unidade principal e as unidades secundárias (CASILLO,2011).

Figura 12 – Comunicação remota entre a unidade principal e as unidades secundárias.



Fonte: Elaborado pelo autor.

As vantagens dos sistemas atuais é que os computadores podem armazenar uma grande quantidade de dados, podendo estes vir de vários lugares diferentes e apresentados ao operador de várias formas. As desvantagens dos sistemas atuais é que se tornaram bem mais complexos, devido a necessidades de conhecimentos específicos, principalmente programação, tanto para o desenvolvimento do sistema quanto para operação do mesmo.

Do ponto de vista do desenvolvedor do *hardware* e *software*, houve uma simplificação no processo de desenvolvimento de dispositivos USB e *softwares* para supervisão, tornando interessante unir as duas tecnologias para desenvolver um sistema de supervisão e controle.

Na questão do desenvolvimento do *hardware*, alguns microcontroladores já possuem o periférico que viabiliza a comunicação USB. Contudo, escrever um *firmware* para este dispositivo era uma tarefa difícil, e para solucionar esse problema, muitos compiladores PICs começaram a incluir bibliotecas que auxiliam no desenvolvimento do protocolo de comunicação e descrição do dispositivo USB, reduzindo bastante os problemas com o desenvolvimento do firmware (MIYADARAI, 2009).

Restando apenas o desenvolvimento do *software* do PC que irá estabelecer comunicação com o dispositivo USB, e para solução deste problema há muito vem sendo desenvolvido bibliotecas e uma estrutura base de código para ser utilizada no *software* Visual Basic, que proporciona a criação da comunicação entre o dispositivo e o *software*. Deve-se ressaltar, que a comunicação USB pode ser dividida em duas classes, a HID e CDC.

A classe de comunicação HID (Human Interface Device), é usada prioritariamente para controlar funções em sistemas de computadores. Os exemplos mais conhecidos são os

teclados, mouses. Os dispositivos que são desenvolvidos utilizando esta classe, são reconhecidos automaticamente pelos atuais sistemas operacionais, ou seja, sem necessidade de instalação de *drivers* ou *software* extras. Por isso, foi escolhido utilizar-se desta classe para o desenvolvimento do presente projeto (ZELENOVSKY; MENDONÇA, 2006).

A classe de comunicação CDC (Connected Device Configuration) é uma classe muito utilizada para transferência genérica de dados, como aqueles presentes em modems e cabos de Ethernet. Contudo, é necessário a instalação de driver adicional para que possa ser estabelecido a comunicação entre o software e o dispositivo. Mais adiante será melhor abordado esta classe de comunicação USB (PINHO, 2010).

Para desenvolvimento do presente trabalho, utilizou-se de uma DLL de 32 bits, ou seja, funciona apenas em aplicativos 32 bits, que irá proporcionar a criação de uma comunicação USB. Então, na plataforma de desenvolvimento Visual Basic é necessário apenas definir VID, PID do dispositivo e o tamanho dos dados que irão trafegar na via de comunicação. Em anexo, neste trabalho, pode-se verificar de forma sucinta, as bibliotecas tanto para desenvolvimento de *hardware*, quanto para *software* (BEKHIT, 2014).

2.10.1. Comunicação USB utilizando classe CDC

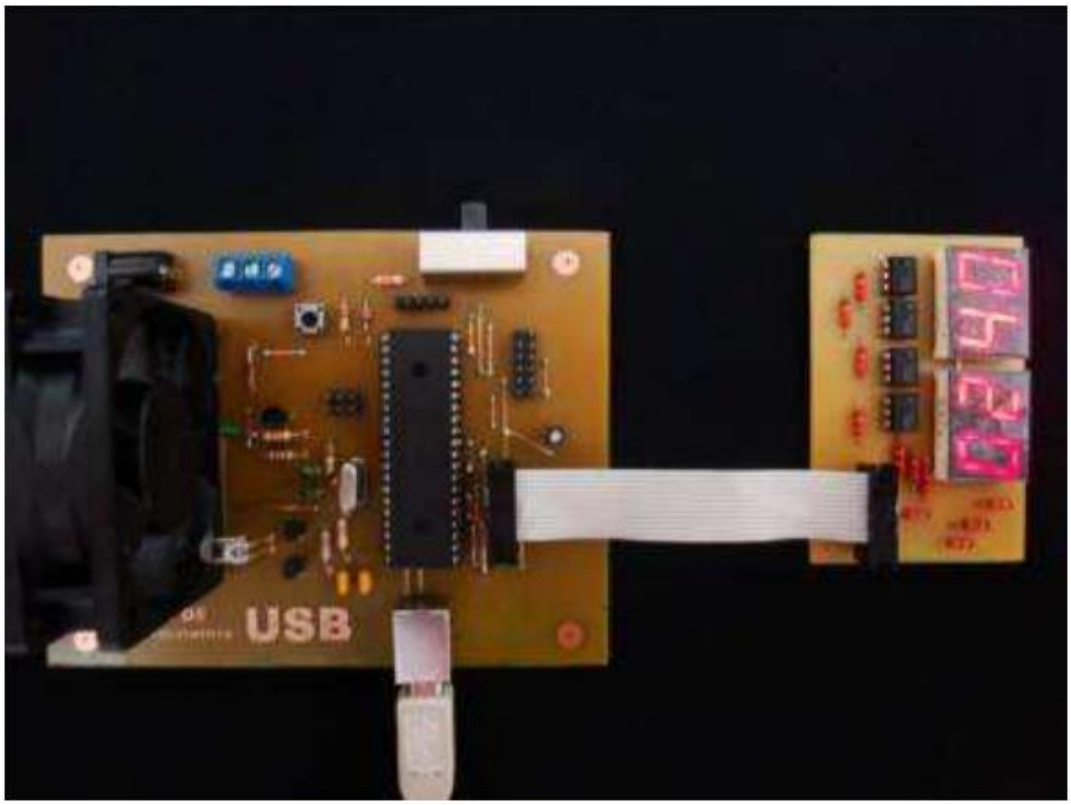
A comunicação USB utilizando-se da classe CDC, em comunicações entre dispositivos onde existe apenas fluxo de dados, não sendo necessário controlar funções do computador.

Para isso, é necessário a instalação de um *driver* no computador, driver este que irá criar uma porta COM serial, apenas virtualmente, que será utilizada para comunicação direta com o dispositivo endereçado para mesma porta. Em outras palavras, o *driver* irá criar um *link* virtual no qual irão se conectar o *software* e o *hardware*.

Por fim, para que o *software* possa comunicar-se com o dispositivo, basta que em seu desenvolvimento possa ser escolhido a porta COM serial que o dispositivo estar acessando, fechando assim o *link* direto de comunicação entre os dois.

No trabalho de conclusão de curso de Germano Bezerra (2010), foi utilizado a classe CDC de comunicação USB para criar um *link* entre *hardware* e o *software* desenvolvido por meio do Matlab, para controlar uma pequena planta composta por um *cooler*, aquecedor, *displays* de 7 segmentos e leitura de conversor A/D. Na figura 13, pode-se ver o dispositivo desenvolvido.

Figura 13 – Dispositivo utilizando classe CDC de comunicação USB.



Fonte: Germano Bezerra, (2010).

2.10.2. Comunicação USB utilizando FT 232

O chipset DT232BM (FTDI, 2014) é uma outra solução utilizada para desenvolver dispositivos que se comuniquem com o computador através do barramento USB, contudo, com velocidade até 3Mbps, através de sinais RS 422 e RS 485 e no máximo de 1Mbps para comunicação através de RS 232. Vale ressaltar que a tecnologia é compatível com os controladores Host USB de versões 1.1 e 2.0.

A vantagem com relação a classe CDC, é que não é necessário que o microcontrolador utilizado possua o periférico de comunicação USB, basta que ele possua o periférico de comunicação serial, RS232.

O fabricante do chipset fornece todos os drivers necessários para o desenvolvimento do hardware e software comunicação. Na figura 14 é ilustrado os tipos de encapsulamento do chipset FT 232 (MESSIAS, 2014).

Figura 14 – Chipset FT 232.

Fonte: FTDI .

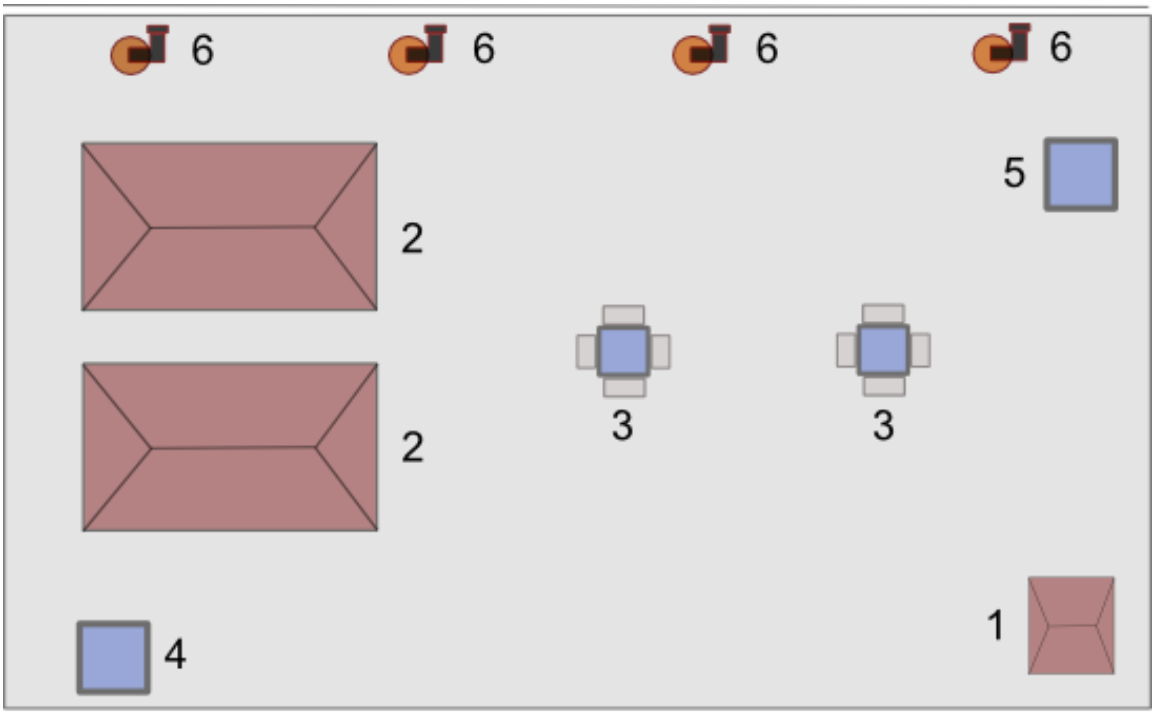
3. Estudo de Caso – Supervisão Sistema Portuário

Tomando como base os métodos e técnicas mais recentes, desenvolveu-se este trabalho, contudo, para análise deste, foi escolhido uma situação real para que possa ser avaliada as potencialidades do projeto. Assim, aplicou-se este a uma planta de um sistema portuário, visando uma comparação com o sistema já utilizado na área portuária de Fortaleza.

3.1. Detalhes do Caso base

Através do sistema empregado no porto, pode-se comandar e supervisionar o sistema de recalque, constituído por bombas e caixas d'água, o sistema de iluminação do porto, constituído por várias torres de iluminação, com um grupo de lâmpadas localizado na parte superior e outro na parte intermediária da torre e o sistema de fornecimento de água para as embarcações, constituído por várias válvulas acionadas por relés. Na figura 15 está representado um mapa ilustrativo de como é o funcionamento da área portuária, contudo, este não representa a realidade.

Figura 15 – Mapa ilustrativo do estudo de caso.



Legenda:
- 1 : Sala de Controle
- 2 : Armazéns
- 3 : Torres de iluminação
- 4 : Reservatório de água inferior
- 5 : Reservatório de água superior
- 6 : Válvula para abastecimento

Fonte: Elaborado pelo autor.

Vale ressaltar os componentes constituintes do sistema real e os *softwares* utilizados para configurar e desenvolver o sistema. Todos os CLPs utilizados, tanto o mestre com os escravos são CLPs da Schneider, sendo o mestre do modelo M340, os utilizados para controle de abastecimento dos navios os OTBs e os utilizados para controle dos postes de iluminação os Twidos.

Com relação aos *softwares* utilizados, o UnityPro foi utilizado para programar o CLP mestre. O Vijeo Citec foi utilizado para criação do supervisório, empregado para mostrar os dados de forma prática para o operador do sistema. E para terminar a lista de *softwares*, foi utilizado o Advantys para programar os CLPs escravos.

Com essas poucas informações até então mostradas sobre o sistema real, pode-se verificar que o sistema aqui desenvolvido tem uma diferença bastante importante com relação a quantidade de *softwares* utilizados, pois utiliza apenas dois, o PIC CCS e o Visual Basic 2010. No caso em particular, como existe apenas uma unidade principal, o PIC CCS foi

utilizado apenas para este dispositivo e caso existissem unidades secundárias, poderia utilizar o mesmo. O Visual Basic 2010 é utilizado para desenvolver o supervisório que irá mostrar os dados para o operador.

Contudo, isso não mostra uma maior ou menor eficiência, só diminui a quantidade de *softwares* pagos utilizados para desenvolver o sistema, já que para sistemas hoje existentes, a quantidade de *softwares* vai depender do esquema utilizado, ou seja, a quantidade e os tipos de CLPs empregados para controlar e supervisionar todo o sistema.

Outra informação importante sobre o sistema base é saber como ocorre a comunicação entre todos os elementos do sistema. A comunicação entre o supervisório e o CLP mestre é feita por Ethernet, como pode-se verificar na figura 16, e entre o CLP mestre e os escravos através de RF, rádio frequência. Sendo a comunicação por RF feita através de RS-485, utilizando-se do rádio XTend-PKG, mostrado na figura 17.

Figura 16 – Representação de cabos de ethernet.

Fonte: Folha Industrial.

Outra diferença importante entre o sistema real e o aqui desenvolvido se mostra no método de comunicação entre o supervisório e a unidade principal. O sistema desenvolvido utiliza comunicação USB, classe HID, entre o PC e a unidade principal, que recebe e envia dados para os outros componentes do sistema, tudo através de cabos. Esta característica é uma desvantagem com relação ao sistema base, devido a possibilidade de erro nas informações ser maior quando realizada através de cabos.

Figura 17 - Rádio XTend-PKG.

Fonte: Elaborado pelo autor.

Contudo, vale ressaltar que o sistema desenvolvido é apenas um protótipo sendo possível desenvolver um protocolo de comunicação mais sofisticado que possibilite o emprego de comunicação por RF, ou até mesmo *wireless*, entre a unidade principal e as unidades secundárias.

3.2. Detalhes do Projeto

O sistema desenvolvido abrange boa parte das funcionalidades do caso base, com apenas algumas diferenças, pois foi escolhido omitir algumas informações, para simplificar o desenvolvimento do protótipo. A principal diferença entre os supervisórios, é o *designer* gráfico, que não foi uma preocupação do presente projeto.

Dos processos supervisionados pelo caso base, o projeto supervisiona apenas o sistema de recalque, verificando os níveis das caixas d'água superior e inferior, contudo, não é possível controlar e verificar o estado das bombas. Outro sistema desenvolvido, foi o de iluminação, onde é possível acionar painéis de iluminação das torres, mas existe uma diferença entre os dois modelos, pois optou-se por criar apenas o painel superior das torres, não existindo nenhum painel na parte intermediária. E por fim, existe também o sistema de abastecimento dos navios, onde é possível comandar e verificar os estados das válvulas de abastecimento de água dos navios.

Um detalhe muito interessante do projeto é que existe a possibilidade de aplicar técnicas de controle que possibilitem uma maior eficiência dos processos, devido a utilização

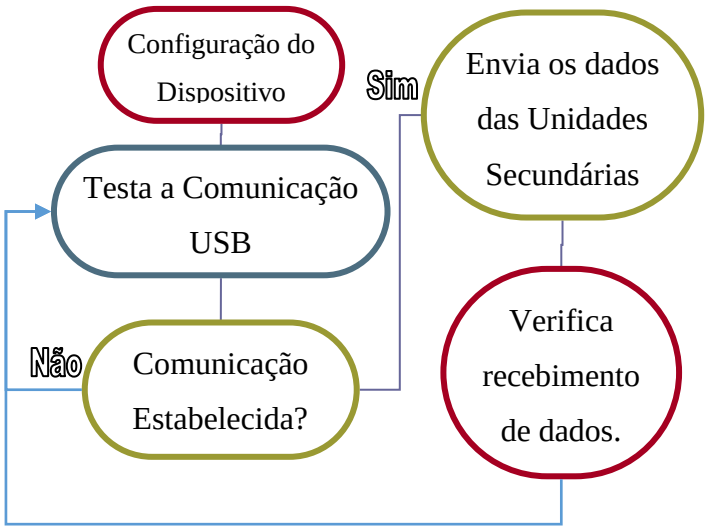
de microcontroladores, desde que possam se comunicar com o *hardware* principal, muito empregados em técnicas de controle discreto.

Por se tratar apenas de um protótipo, optou-se fazer essas simplificações para que possa ser feito uma análise melhor das grandes possibilidades de melhoria e aplicações do projeto.

4. Descrições do Dispositivo de Supervisão e Controle

O dispositivo de supervisão e controle, DSC, tem como objetivo estabelecer comunicação com o *software* por meio de comunicação USB. Todos os comandos enviados e recebidos são processados pelo próprio dispositivo, não havendo unidades secundárias, existindo apenas a unidade principal. Para melhor entender processamento dos dados, desde a conexão com o computador até a troca de dados com o supervisório, pode-se analisar o diagrama de blocos da figura 18.

Figura 18 – Diagrama de blocos do funcionamento do DSC.



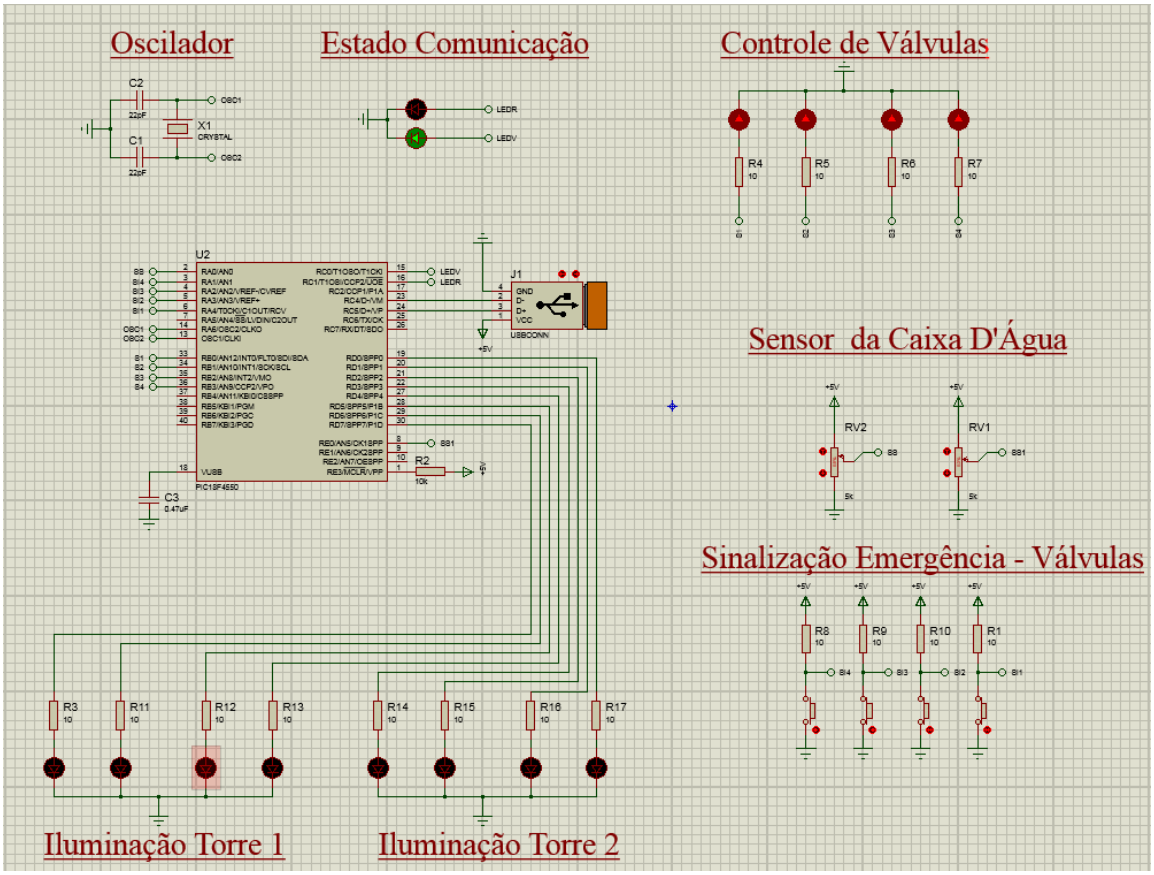
Fonte: Elaborado pelo autor.

Para dar sequência na melhoria do projeto, se faz necessário desenvolver um protocolo de comunicação que permita a existência de unidades secundárias e que esta esteja adequada a uma comunicação via rádio frequência, RF. Contudo, atualmente já existe a possibilidade de utilizar módulos que permitam a comunicação via *wireless*, caso exista no ambiente.

Na figura 19, pode-se ter uma visão global de todo o projeto, entretanto, é apenas uma representação simplificada, sendo necessário circuitos auxiliares para processar os dados e

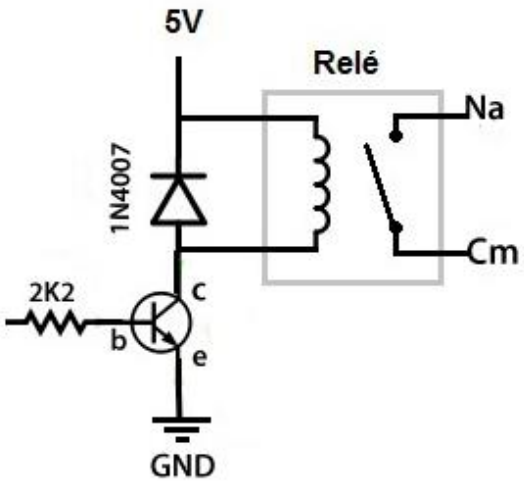
executar o comando de acionamento dos relés. Na figura 20, é representado o circuito mais simples para acionamento de um relé.

Figura 19 – Visão geral da unidade principal e suas funções.



Fonte: Elaborado pelo autor.

Figura 20 – Esquema de acionamento de um relé.



Fonte: Elaborado pelo autor.

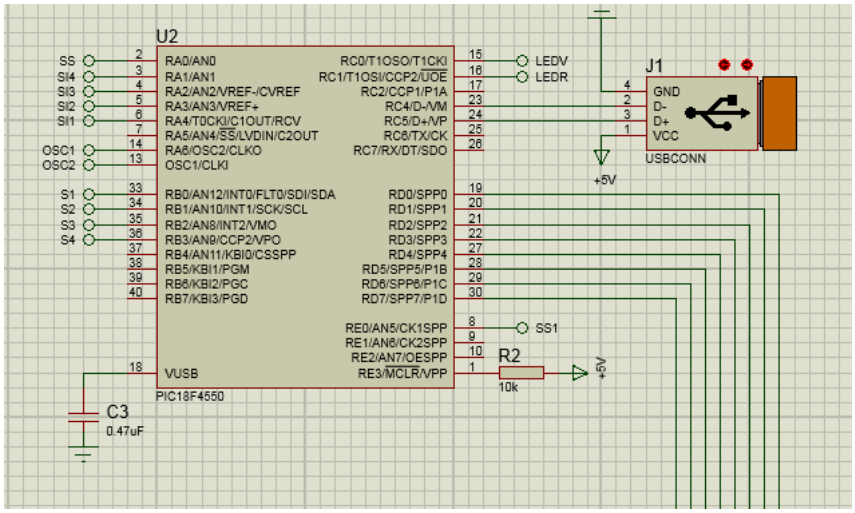
4.1. Unidade Principal do DSC– Protocolo de Comunicação

Como já foi mencionado, o dispositivo de supervisão e controle ou DSC, é o dispositivo desenvolvido para estabelecer comunicação com o software. O hardware é composto de 3 partes principais, o microcontrolador, oscilador e sinaleiras para indicar o estado da comunicação. Na figura 21, pode-se verificar o microcontrolador utilizado para se comunicar com o software.

Vale ressaltar que todo o protocolo de comunicação é desenvolvido no microcontrolador, contudo, deve estar compatível com o a programação desenvolvida no supervisório do computador.

Outro detalhe importante, é que o DSC utiliza o modo de funcionamento *Self-Powered* sendo necessário o projeto de uma fonte de alimentação para o circuito. Posteriormente, será necessário prever uma fonte de alimentação para cada unidade secundária. Como já foi mencionado, este tipo de operação é o desejado, pois desta forma o projeto não será limitado por corrente máxima.

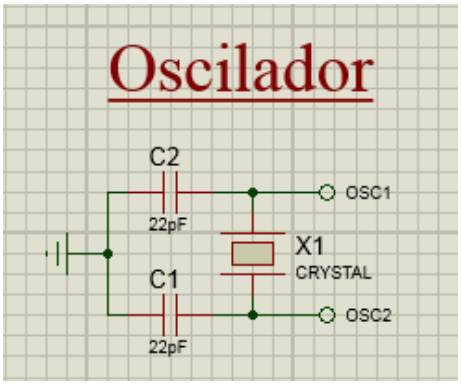
Figura 21 – Microcontrolador utilizado para estabelecer a comunicação com o supervisório.



Fonte: Elaborado pelo autor.

A segunda parte da unidade principal do DSC é o oscilador, elemento fundamental para garantir qualidade na comunicação. Na figura 22, pode-se verificar o esquema de ligação do oscilador.

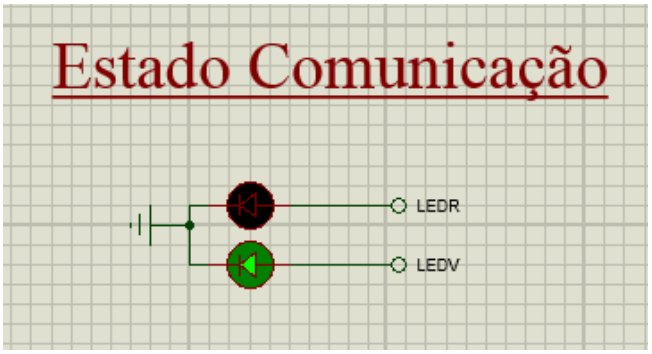
Figura 22 – Oscilador do DSC.



Fonte: Elaborado pelo autor.

A última parte da unidade principal do DSC é a sinaleira de estado da comunicação, onde será indicado com sinaleira vermelha se o estabelecimento de comunicação entre o DSC e *software* ainda não occorreu. A indicação com sinaleira verde indica que a comunicação já foi estabelecida e já é possivel o trafego de informações. Esta sinalização existe tanto no *hardware* como no próprio supervisório. Na figura 23 é representado estas sinaleiras.

Figura 23 – Sinaleira de estado do DSC.



Fonte: Elaborado pelo autor.

4.2. Unidades Secundárias do DSC – Sensores e relés

As unidades secundárias tem como objetivo executar comandos ou coletar dados de sensores. Contudo, neste trabalho o desenvolvimento destas unidades não foi totalmente desenvolvido, por se tratar de um protótipo, desta forma adotou-se apenas formas simples e funcionais para aplicação em bancada, para que fosse observado o funcionamento. Porém, as atividades de cada unidade serão abordadas de forma específica.

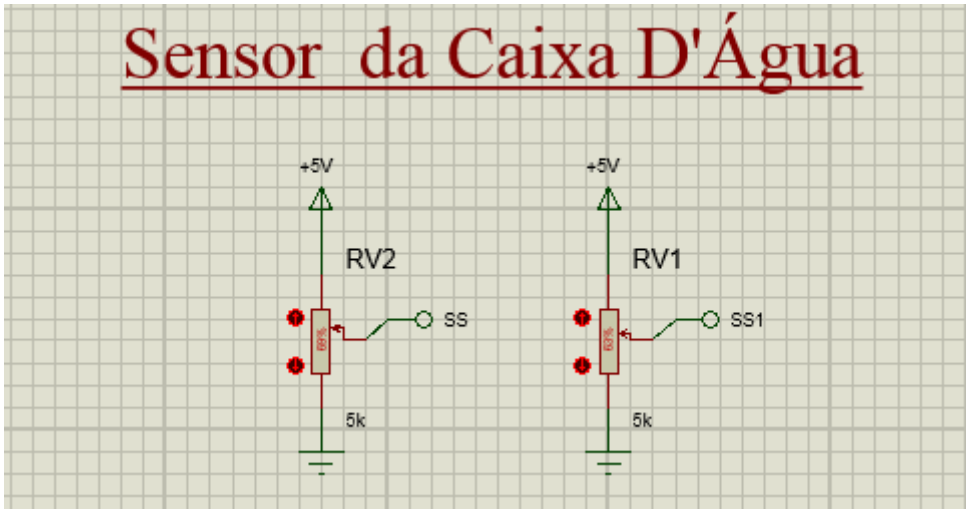
4.2.1. Sensores para Caixa D'Água

Os sensores das caixas d'Água são utilizados para verificar o nível das caixas, para que não venha a faltar, e as atividades possam ser atendidas adequadamente, principalmente o abastecimento de água dos navios.

O sensor possui o mesmo funcionamento de um potênciometro, pois desta forma será possível verificar a porcentagem exata de volume de água existente dentro dos reservatórios. À medida que o reservatório enche ou esvazia, o sensor enviará para sua saída uma tensão entre 0 e 5 volts, sendo que o valor 0 significa que o reservatório estar completamente vazio e 5 volts representa o reservatório completamente cheio. Pode-se verificar o modelo de sensor utilizado para a simulação na figura 24.

Para o protótipo aqui apresentado, o DSC irá receber esses valores de tensão e irá fazer a conversão no A/D e em sequida enviará para o supervisório, no computador, o valor em *Bytes*. Todavia, para projetos futuros, o ideal seria que a conversão fosse feita na própria unidade *secudária* e que fosse enviado apenas o valor em *bytes* da conversão. Para que isso seja possível, é necessária a implemetação de um protocolo de comunicação mais sofisticado, como já foi mencionado, em outras palavras que possibilite essa troca de dados.

Figura 24 – Sensores da caixa D'Água inferior e superior.



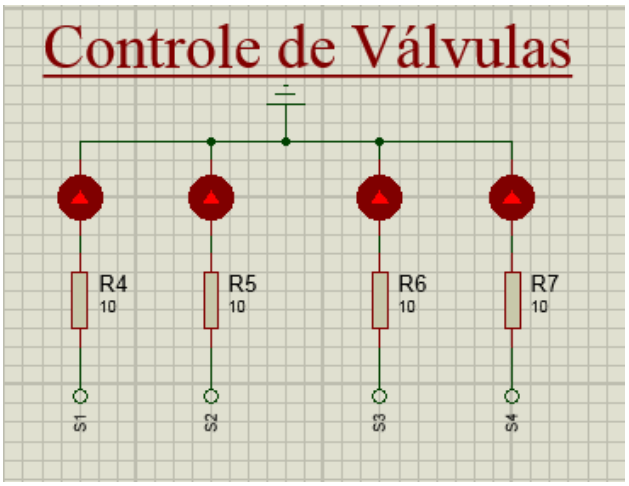
Fonte: Elaborado pelo autor.

4.2.2. Relés para acionamento das válvulas

O circuito de comando para as válvulas é utilizado para abastecimento dos navios, sendo este comandado através do *software* da sala de controle. O esquema apresentado na figura 25 é apenas ilustrativo, apenas para verificar o funcionamento do protótipo, pois para acionamento de um relé, é necessário um circuito auxiliar, como o que já foi mostrado na figura 20.

Um detalhe interessante é a possibilidade de utilização de microcontroladores, que através de técnicas de controle possam aumentar a eficiência da atividade, onde o abastecimento possa ser feito no menor tempo possível, por exemplo, além de permitir a implementação de um protocolo de comunicação.

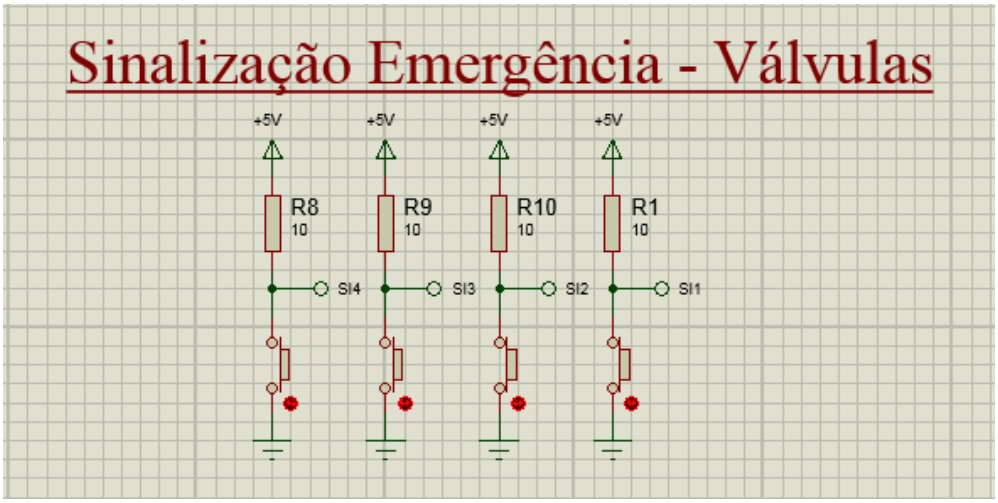
Figura 25 – Representação de relés para acionamento das válvulas.



Fonte: Elaborado pelo autor.

Uma medida de proteção importante adotada no projeto foi a implementação de comando manual junto a válvula, utilizada apenas em casos de emergência, caso haja falha no comando remoto. Este comando manual irá sinalizar no supervisório do computador que algo ocorreu e, desta forma, o operador irá até o local verificar o problema. Pode-se verificar a ilustração do comando de emergência na figura 26.

Figura 26 – Comando para emergências.



Fonte: Elaborado pelo autor.

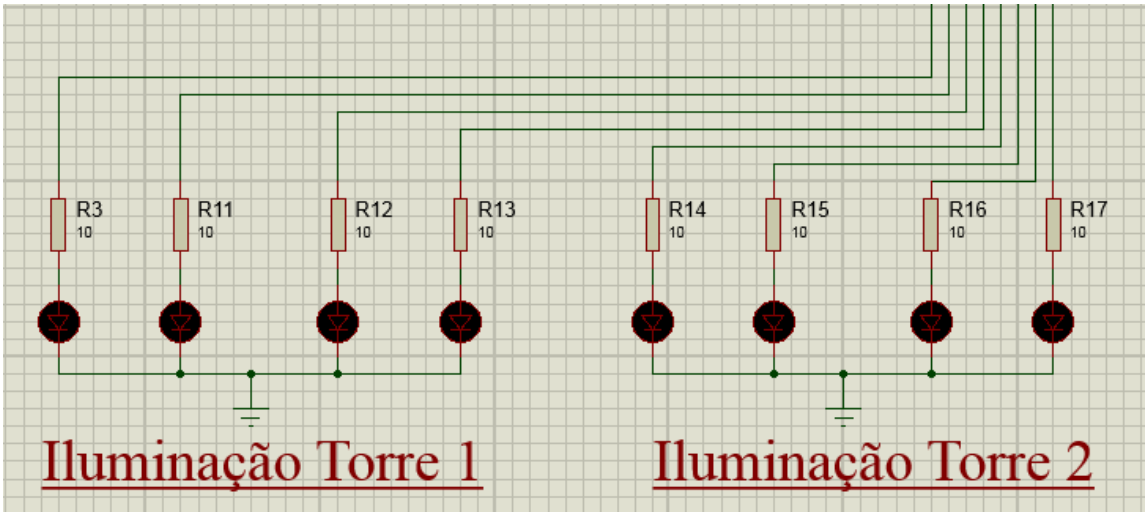
4.2.3. Relés para acionamento da Iluminação

Para acionamento das torres de iluminação, é necessária a utilização de relés, seguindo o mesmo princípio de funcionamento das válvulas de abastecimento. O esquema utilizado em simulação para verificar o funcionamento está ilustrado na figura 27.

Com os avanços da tecnologia de iluminação, é possível hoje em dia dimerizar a luz através da utilização de PWM (Pulse-Width Modulation), permitindo assim uma utilização mais eficiente da iluminação. Então, com a utlização de microcontrolador nessas unidades secundárias, é possível implemetar mais facilmente essas melhorias.

Vale ressaltar que cada torre de iluminação é composta por 4 zonas de iluminação, e cada uma destas é comandada por um relé. Entretanto, para utilização da técnica de dimerização através de chaveamento por PWM, se faz necessário a utilização de outros componentes que permitam esse chaveamento, pois o relé poderá ser danificado rapidamente devido frequência de chaveamento. Um dos componentes mais utilizados atualmente para chaveamento são os IGBTs, mostrado na figura 28.

Figura 27 – Esquema utilizado para ilustrar as torres de iluminação.



Fonte: Elaborado pelo autor.

Figura 28 – Componente de chaveamento, IGBT.

Fonte: Elaborado pelo autor.

5. Supervisório do DSC

O *software* de supervisão, foi desenvolvido no Visual Basic 2010 Express, pois é uma plataforma que fornece uma área de desenvolvimento mais visual, como próprio nome já diz, o que facilita muito a criação da parte gráfica do software.

Para a criação do supervisório, é necessária a utilização de uma DLL para sistemas Windows X86, considerando que neste projeto foi utilizado a classe HID de comunicação USB, no qual estão todas as funções necessárias para que possa ser estabelecida a comunicação entre DSC e o supervisório do mesmo. Contudo, ao se iniciar o desenvolvimento, é necessário adicionar um arquivo de código já desenvolvido, onde todas as funções da DLL são iniciadas e estarão disponíveis para utilização durante todo o código.

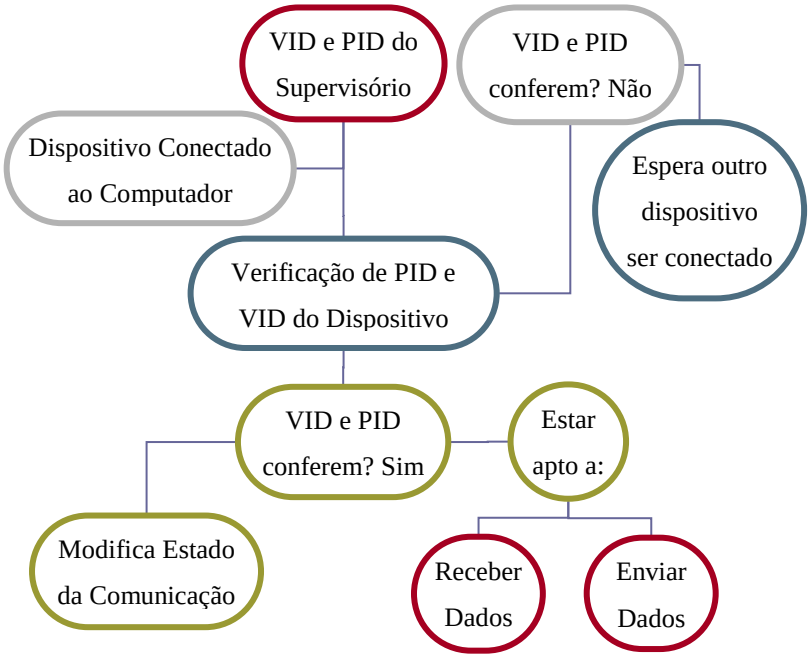
O *software* é composto por 7 partes, sendo estas:

- Tela de Bloqueio
- Tela Principal
- Tela Sobre
- Tela do Manual
- Tela Modo Abastecimento
- Tela Modo Recalque
- Tela Modo Iluminação

Antes de visualizar a parte gráfica, se faz necessário entender o protocolo e as rotinas utilizadas para criar o *link* entre o supervisório e o DSC. Para isso, pode-se entender melhor pelo diagrama de blocos da figura 29. Fazendo o acompanhamento pelo diagrama de blocos, chega-se à conclusão que trata-se de uma rotina de programação simples, sendo necessário apenas identificar as funções para cada ação.

De imediato, pode-se notar a diferença fundamental entre a classe CDC e HID de comunicação USB, pois através do HID não é preciso habilitar a conexão no próprio supervisório, basta conectar o dispositivo ao computador. Na comunicação CDC, é necessário escolher a porta de comunicação serial, virtual, com a qual deseja se comunicar.

Figura 29 – Diagrama de blocos da programação do supervisório.


VID e PID do Supervisório
VID e PID conferem? Não
Dispositivo Conectado ao Computador
Espera outro dispositivo ser conectado
Verificação de PID e VID do Dispositivo
VID e PID conferem? Sim
Estar apto a:
Modifica Estado da Comunicação
Receber Dados
Enviar Dados

Fonte: Elaborado pelo autor.

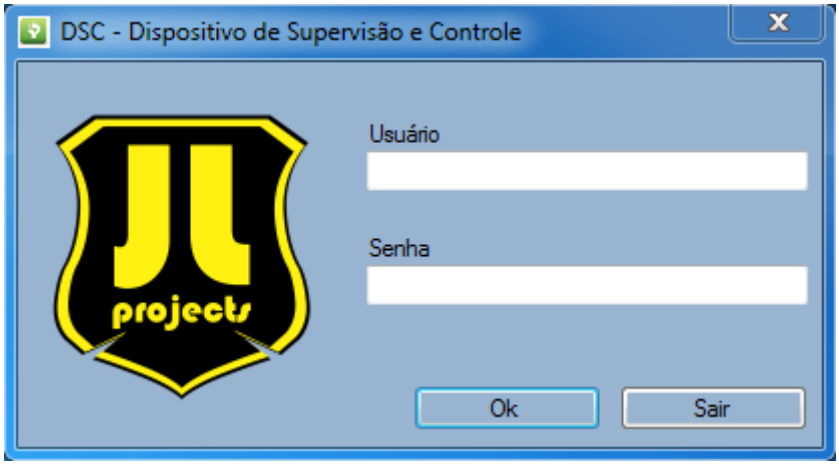
A rotina para que a comunicação USB seja estabelecida é simples, contudo, a medida que sistemas mais complexos vão sendo criados, o processamento das informações deve ser tratado com cautela, devido ao aumento do número de variáveis. O aumento do número de variáveis a serem monitoradas ou controladas estão diretamente relacionadas com o tamanho dos dados que podem trafegar por vez, no caso um byte por vez.

Desta forma, para valores que necessitem de dois bytes, como em conversores A/D, deve-se desenvolver um protocolo de tratamento dos dados mais elaborado, de modo a atender as necessidades do projeto.

5.1. Tela de Bloqueio

A tela de bloqueio foi desenvolvida para que apenas pessoas autorizadas possam utilizar o software, sendo as informações de usuário e senha informadas na aquisição do dispositivo. Algo que pode ser melhorado, é a possibilidade de alterar o usuário e senha, o que não é possível no atual protótipo. Pode-se verificar a tela de bloqueio na figura 30.

Figura 30 – Tela de Bloqueio.



Fonte: Elaborado pelo autor.

Caso o operador do sistema coloque uma das informações erradas, irá aparecer uma tela de aviso informando que umas das informações apresentadas está errada. Isto não acarretará em uma proteção especial, contudo, pode-se adicionar mais está opção para versões futuras do sistema.

A logo mostrada na figura 28 é apenas de caráter ilustrativo, já que até o momento não havia sido iniciado os processos legais para abertura de qualquer empresa.

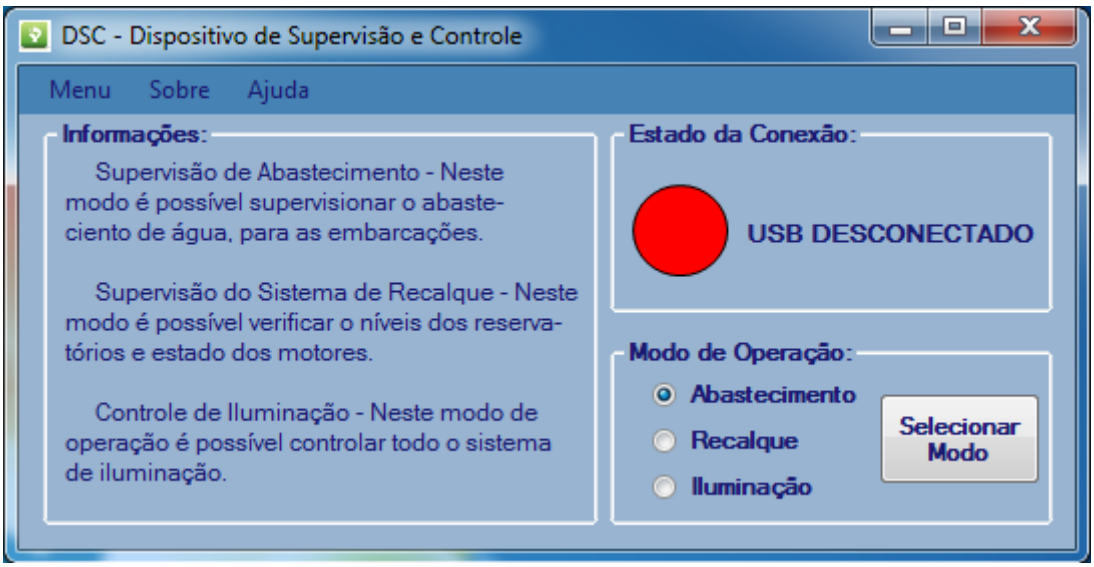
5.2. Tela Principal

Pode-se visualizar a tela principal na figura 31. Na tela principal é onde se encontram todas as opções para o uso do equipamento. Na aba superior de menus encontra-se três opções:

- Menu: Onde existe a opção de bloquear o software, sendo transferido para a tela de bloqueio, e opção de sair do supervisório, que irá desligá-lo.
- Sobre: É onde se encontra informações sobre quem desenvolveu o dispositivo aqui utilizado e qual os objetivos do equipamento.
- Ajuda: Neste menu encontra-se informações úteis para a boa utilização do sistema.

Logo abaixo dos menus existe uma caixa com informações sobre cada modo de operação do sistema, o que não dispensa a leitura do manual. Restando apenas mais dois blocos, um deles contêm informações sobre o estado da comunicação, ficando a sinaleira verde se a comunicação foi estabelecida, caso contrário a sinaleira estará vermelha, e o último bloco seria para selecionar o modo de operação que desejar utilizar.

Figura 31 – Tela principal.


DSC - Dispositivo de Supervisão e Controle
Menu
Sobre
Ajuda
Informações:
Supervisão de Abastecimento - Neste modo é possível supervisionar o abastecimento de água, para as embarcações.
Supervisão do Sistema de Recalque - Neste modo é possível verificar o níveis dos reservatórios e estado dos motores.
Controle de Iluminação - Neste modo de operação é possível controlar todo o sistema de iluminação.
Estado da Conexão:
USB DESCONECTADO
Modo de Operação:
Abastecimento
Recalque
Iluminação
Selecionar Modo

Fonte: Elaborado pelo autor.

5.3. Tela Sobre

Na figura 32, pode-se verificar a tela sobre e ler todas as informações apresentadas. A tela sobre nada mais é do que uma página com informações sobre o desenvolvedor do projeto e quais são os objetivos do projeto. A informações finais são apenas de caráter autoral, e para lembrar dos direitos de propriedade intelectual.

A logo e nome de empresa, JL Automotion and Eletric Project, como já havia sido mencionado são apenas de caráter ilustrativo, já que até o presente momento não havia sido iniciado os processos legais para abertura de uma empresa, contudo, optou-se por esta ilustração para atribuir ao projeto um caráter mais profissional.

Na parte inferior da tela pode ser visualizado um botão, que tem como função fechar a tela sobre, contudo, isto não finaliza o programa, apenas fecha a tela em questão.

Figura 32 – Tela sobre.

Sobre
JL Automation and Eletric Projects
Sobre DSC:
Projeto desenvolvido pelo Engenherio Eletricista, Junior Landim, tem como objetivo, criar um novo dispositvo que possa fornecer soluções de automação para projetos elétricos, desde resisdências até grandes indústrias.
Copyright 2014 JL Automation and Eletric Projects. Todos os direitos reservados. O equipamento e sua interface do usúario são protegidos por marca registrada e outros direitos de propriedade intelectual existentes ou pendentes no Brasil e em outros países.
Sair

Fonte: Elaborado pelo autor.

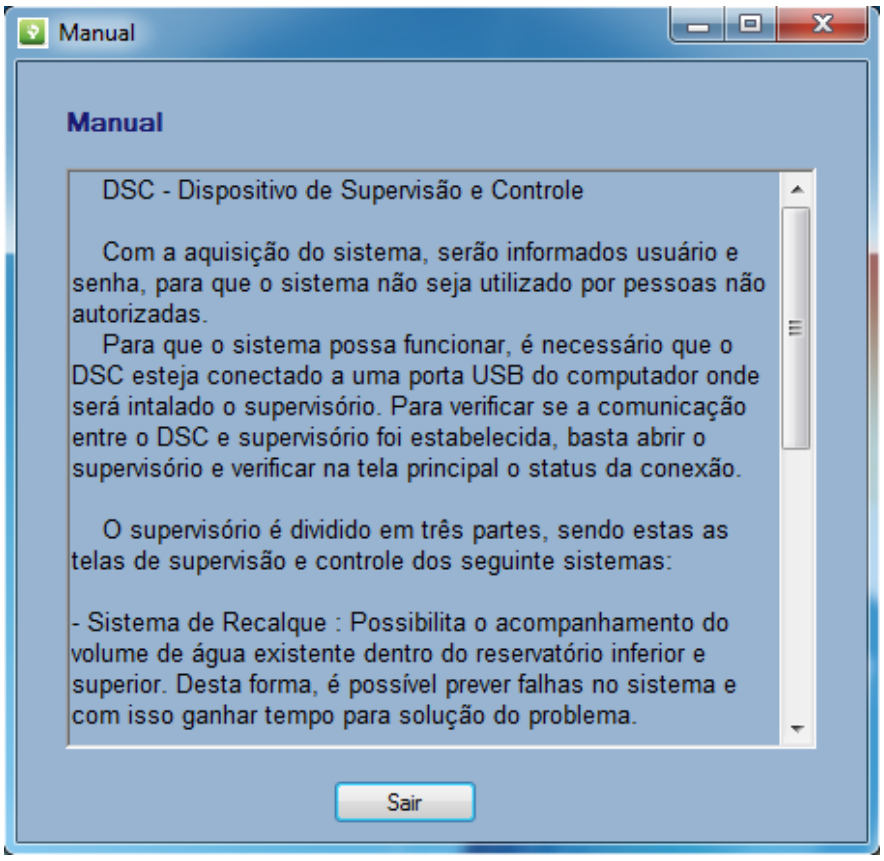
5.4. Tela do Manual

A tela manual tem como objetivo fornecer informações sobre o sistema, tais como, o modo de funcionamento do sistema, quais são os modos de operação do sistema, como selecionar cada um e quais os objetivos de cada um deles.

As informações contidas no manual auxiliam o usuário a entender e interpretar todas as sinaleiras e botões existentes no supervisório, para que ele possa operar o sistema de forma adequada e sem operações equivocadas.

Contudo, por se tratar de um protótipo o manual do usuário ainda estar em desenvolvimento, sendo bastante sucinto, mas apresenta todas as informações necessárias para utilização do protótipo. Este tipo de material, com base em pesquisas e em experiências de usuários, é passível de constantes atualizações já que os usuários sempre encontram dificuldades que podem não se encontrar no manual ou simplesmente não foram bem explicitadas. Sendo assim, é necessário um continuado *feedback* dos usuários que utilizam o sistema. Na figura 33, pode-se visualizar a tela do manual.

Figura 33 – Tela do manual.


Manual
Manual
DSC - Dispositivo de Supervisão e Controle
Com a aquisição do sistema, serão informados usuário e senha, para que o sistema não seja utilizado por pessoas não autorizadas.
Para que o sistema possa funcionar, é necessário que o DSC esteja conectado a uma porta USB do computador onde será intalado o supervisório. Para verificar se a comunicação entre o DSC e supervisório foi estabelecida, basta abrir o supervisório e verificar na tela principal o status da conexão.
O supervisório é dividido em três partes, sendo estas as telas de supervisão e controle dos seguinte sistemas:
- Sistema de Recalque : Possibilita o acompanhamento do volume de água existente dentro do reservatório inferior e superior. Desta forma, é possível prever falhas no sistema e com isso ganhar tempo para solução do problema.
Sair

Fonte: Elaborado pelo autor.

No manual redigído na figura 31, encontra-se as seguinte informações:

‘’ *DSC - Dispositivo de Supervisão e Controle*

Com a aquisição do sistema, serão informados usuário e senha, para que o sistema não seja utilizado por pessoas não autorizadas.

Para o bom funcionamento do sistema, é necessário que o DSC esteja conectado a uma porta USB do computador onde será instalado o supervisório. Para verificar se a comunicação entre o DSC e supervisório foi estabelecida, basta abrir o supervisório e verificar na tela principal o status da conexão.

O supervisório é dividido em três partes, sendo estas as telas de supervisão e controle dos seguintes sistemas:

- Sistema de Recalque: Possibilita o acompanhamento do volume de água existente dentro do reservatório inferior e superior. Desta forma, é possível prever falhas no sistema e com isso ganhar tempo para solução do problema.

- Sistema de Abastecimento: Por meio de botões e sinaleiras é possível controlar o abastecimento de água dos navios. As sinaleiras presentes indicam o status das válvulas, devido a possibilidade de fecha-las de forma manual, ficando assim interrompido o controle remotamente.

- Sistema de Iluminação: É composto de botões e sinaleiras, que podem controlar remotamente as torres de iluminação. Sendo estas torres divididas em quatro zonas, diminuindo assim o desperdício de energia elétrica.

É importante lembrar que o sistema foi desenvolvido para atuar no atual esquema apresentado, caso haja modificações, é necessário um retrofit do sistema automático para adicionar novas funcionalidades e garantir o bom funcionamento.

A tecnologia aqui utilizada, tanto para hardware como software, é de propriedade intelectual do Eng. Eletricista Júnior Landim, ficando assim proibido a reprodução não autorizada previamente. ’’

5.5. Tela Modo Abastecimento

Na tela de abastecimento, pode-se verificar sinaleiras e botões que possibilitam o controle e supervisão do abastecimento de água dos navios.

No primeiro quadro, estado das válvulas, pode-se controlar e verificar o estado de cada válvula de abastecimento. Já no segundo quadro, trata-se de sinaleiras que tem como função identificar o acionamento do botão de emergência que fica junto a válvula.

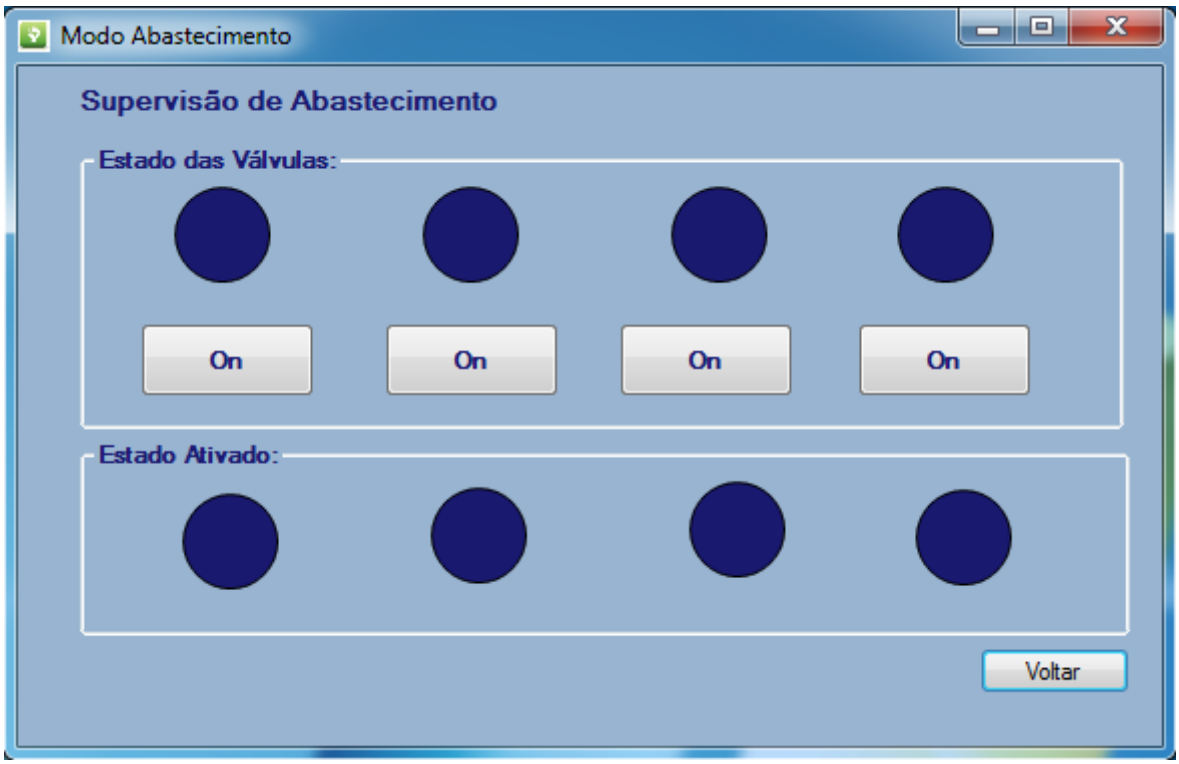
A figura 34 mostra a tela modo de abastecimento, verificando-se um layout simples. Como já mencionado, a preocupação com o *designer* não foi o foco do trabalho aqui apresentado.

É sabido que este modo de operação tem programação simples, visando apenas a visualização da comunicação entre o DSC e supervisório. Contudo, para versões posteriores do equipamento seria interessante, utilizar mais linhas de código que visem a proteção do sistema, como a impossibilidade de clicar em um botão o qual o equipamento vinculado esteja em situação de emergência.

Como já mencionado algumas vezes no corpo deste trabalho, a utilização de microcontroladores, nas unidades secundária, em versões posteriores do sistema aqui utilizado seria muito interessante, pois desta forma poderia ser controlado a vazão e volume de água, podendo até mesmo ser automaticamente encerrado o processo de carga, caso um volume predeterminado já tenha sido atingido. Contudo, para que esses parâmetros fiquem dentro de limites aceitáveis, é necessário utilizar-se de técnicas de controle discreto.

No entanto, para que isso seja possível, mais uma vez é notável a necessidade da criação de protocolo de comunicação mais sofisticado, pois se torna necessário enviar e receber mais dados controle.

Figura 34 – Tela de abastecimento.



Fonte: Elaborado pelo autor.

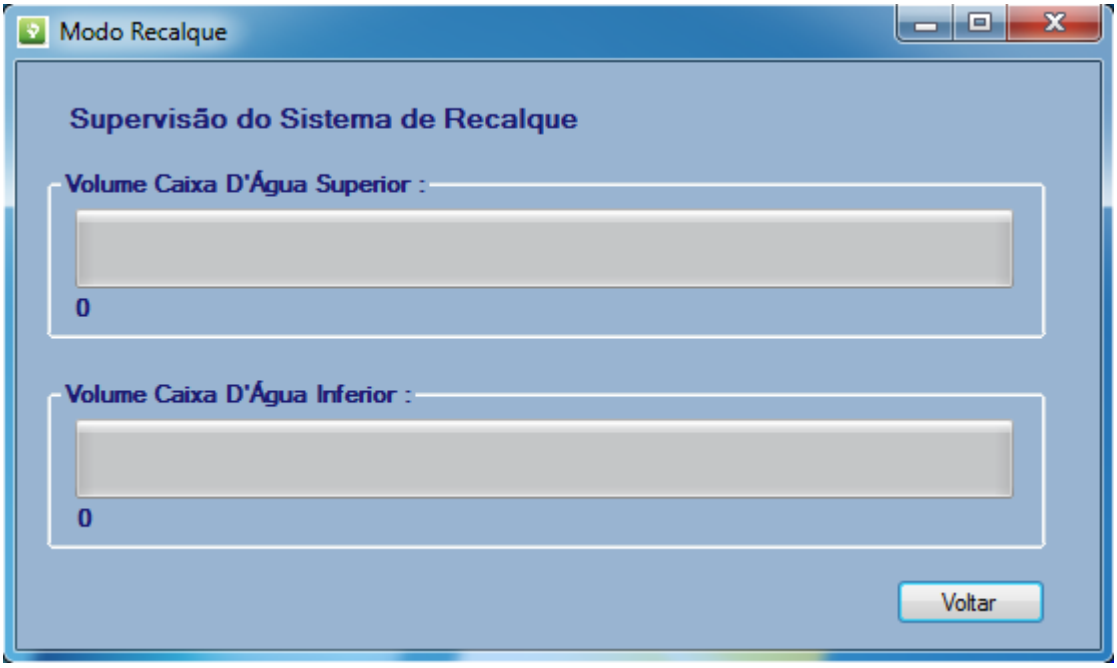
5.6. Tela Modo Recalque

A tela em questão é utilizada para controlar e supervisionar o sistema de recalque, principalmente para verificar os níveis dos reservatórios, superior e inferior. Nenhuma outra funcionalidade foi integrada a este modo de funcionamento, logo a supervisão é feita somente através de sensores que mandam a informação do nível do reservatório. A figura 35, mostra a tela do modo de recalque.

Contudo, outras funções poderiam ser integradas para versões futuras do protótipo, como, controle de vazão para horários em que o consumo de água esteja elevado, podendo ser compensado e não prejudicando o funcionamento das atividades que necessitam de água. Outra função que poderia ser adicionada seria botões para controle das bombas, sendo possível ativá-la e desativá-la remotamente.

Vale ressaltar que se o reservatório inferior fique abaixo de um valor crítico, a bomba deverá ser desativada automaticamente, pois esta não pode operar sem água, caso contrário irá queimar. Esta função pode ser integrada ao sistema ou feita através de circuito de comando. Sendo uma boa opção a utilização de quadro de comando próximo as bombas, para a atual versão do sistema.

Figura 35 – Tela de recalque.



Fonte: Elaborado pelo autor.

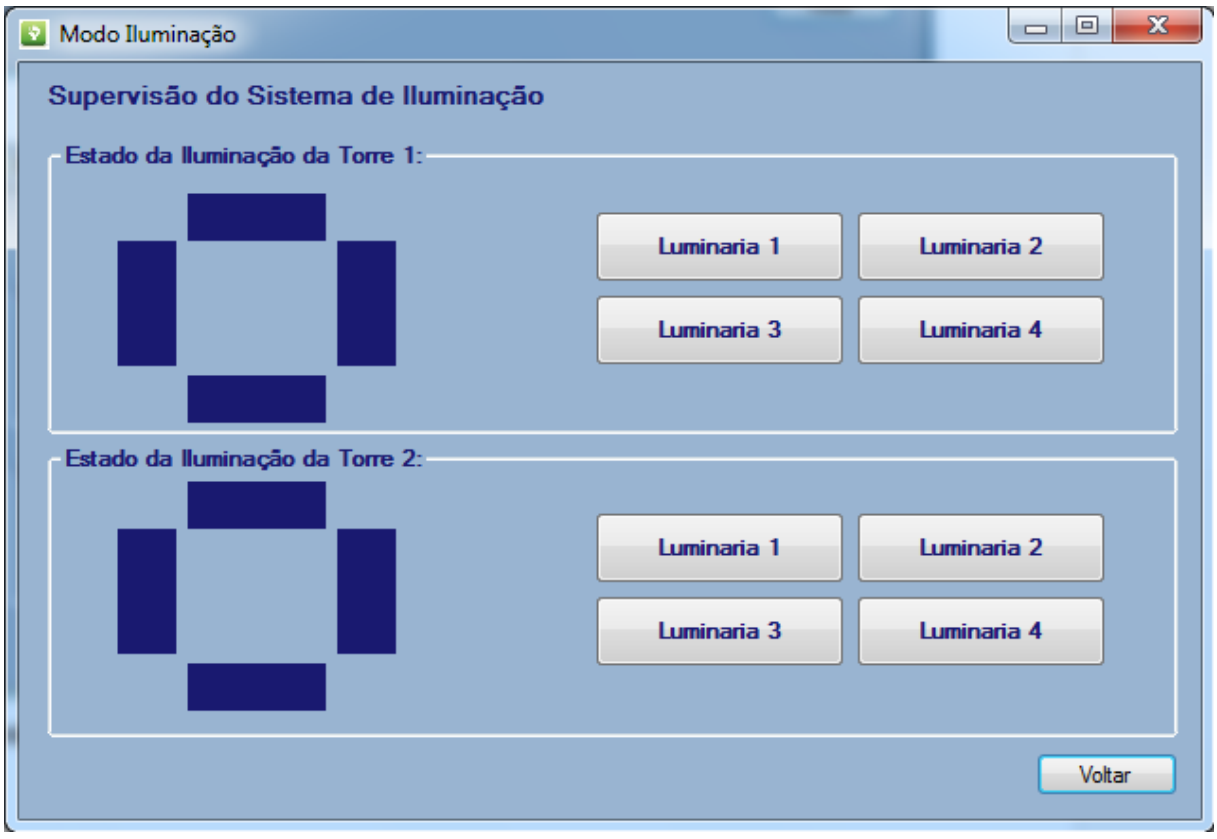
5.7. Tela Modo Iluminação

A criação deste modo de operação confere ao sistema uma maior eficiência energética, pois trata-se de uma iluminação de potência elevada, cuja execução de tarefas que necessitam de um amplo campo de visão.

Na figura 36 é possível observar duas torres de iluminação, que são divididas em 4 zonas. A criação dessas zonas é o que confere uma maior eficiência para o sistema, pois ao invés de ligar toda a iluminação, apenas ativa-se as zonas que garantam a segurança das atividades que estão sendo executadas na área.

A planta de ilustração do estudo de caso é mostrada na figura 15, na qual observa-se que as torres estão localizadas na parte central da planta em estudo. Para exemplificar e facilitar o entendimento da eficiência das zonas de iluminação, considera-se que algum serviço esteja sendo executado próximo as válvulas para abastecimento, então, para garantir uma boa visibilidade para esses serviços, ativa-se apenas as luminárias voltadas para esta área, permanecendo todas as outras desligadas.

Figura 36 – Tela de iluminação.

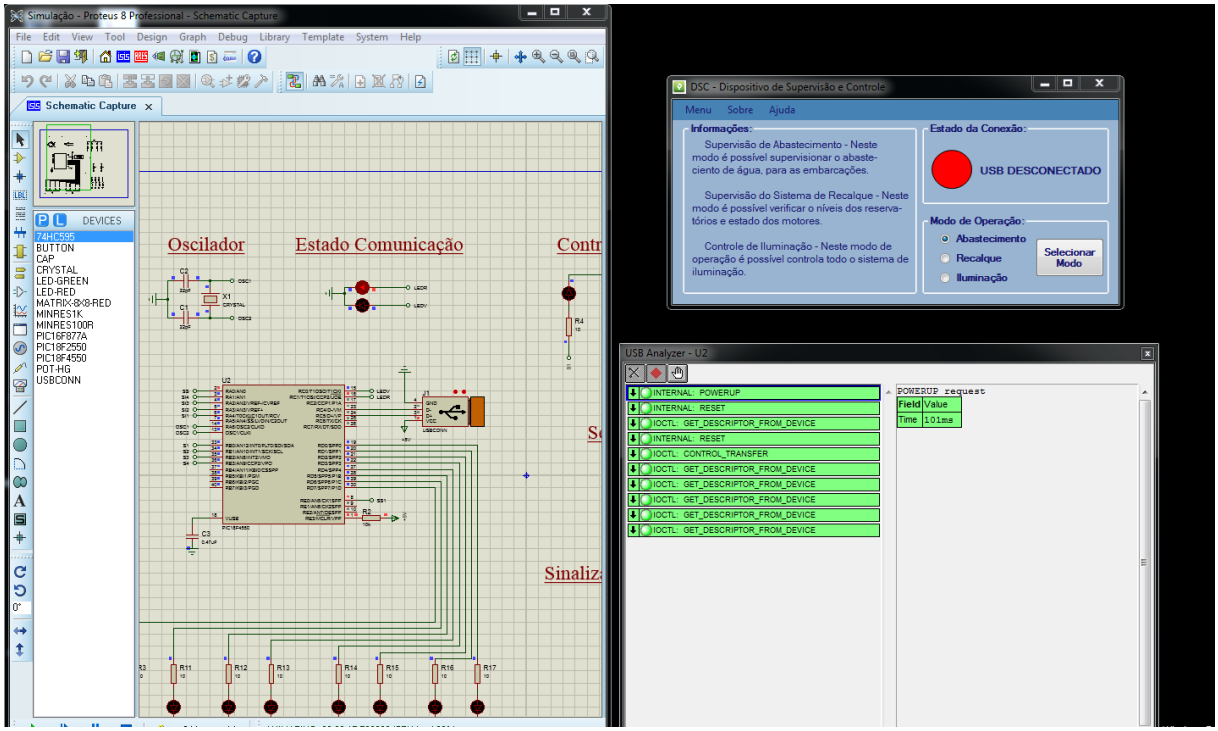


Fonte: Elaborado pelo autor.

6. Resultados de Simulação

Até o momento, o protótipo foi apenas simulado, mostrado na figura 37. Todos os testes de simulação foram feitos utilizando o software Proteus, da empresa Labcenter Electronics Ltd, que possui uma expansão onde é possível ativar o periférico USB do microcontrolador e desta forma estabelecer uma comunicação entre o supervisório e o DSC, sendo o hardware no caso apenas virtual.

Figura 37 – Simulação da comunicação entre o supervisório e o DSC.



Fonte: Elaborado pelo autor.

Todos os resultados obtidos através de simulação foram satisfatórios. Contudo, nos primeiros testes executados, uma pequena falha foi observada, sendo esta um atraso entre o envio dos dados enviados entre o microcontrolador e o supervisório, obedecendo esta direção de dados. No fluxo inverso de dados, supervisório para o microcontrolador, não existe esta falha, sendo feita a comunicação normalmente.

O problema, de imediato, se tornou um empecilho para o desenvolvimento do projeto, pois não era possível desenvolver nenhum tipo de protocolo para receber os dados do microcontrolador, estando a comunicação totalmente fora de sincronismo. *A priori* foi difícil identificar o erro, mas como a comunicação entre o microcontrolador e supervisório estava

ocorrendo, apesar de fora de sincronismo, foi então proposto colocar um atraso na rotina do microcontrolador, solucionando o problema, e possibilitando desenvolvimento de um protocolo de comunicação.

Apesar de solucionado o principal problema encontrado, outro entrave surgiu, que seria o atraso gerado, propositalmente, entre os dados recebidos pelo supervisório. Surgindo assim a principal desvantagem do projeto aqui desenvolvido, apesar de se tratar de um atraso de apenas 300ms. Todavia, para projetos mais complexos este atraso se tornaria um problema bem maior.

Para versões futuras dos projetos, propõe-se criar um protocolo de comunicação que permita a utilização de comunicação sem fio, rádio frequência, entre a unidades secundárias e a unidade principal. Assim, seria agregado mais um atraso devido a comunicação sem fio, e o processamento da informação pelas unidades secundárias, tornando-se crítico a questão do tempo para sistemas mais complexos.

Contudo, acredita-se que este atraso se deve ao fato de se tratar de uma simulação, sendo assim possível que o atraso não exista no protótipo real. No entanto, trata-se de uma especulação, não sendo confirmado com o protótipo real.

Para que a comunicação entre o microcontrolador e o hardware fosse estabelecida, apenas duas informações deviam ser alteradas nos dois códigos, o VID e o PID, pois estas serão as informações verificadas pelo supervisório, para perceber que DSC foi conectado ao computador.

Para dar seguimento ao trabalho faz-se necessário a montagem do protótipo real e verificar os problemas que serão encontrados. E desta forma continuar promovendo melhorias para o projeto.

7. Conclusões

O trabalho desenvolvido teve resultados bastante satisfatórios, restando apenas um entrave com relação ao atraso surgido para sanar a falta de sincronismo no fluxo de informações do microcontrolador para o computador.

O projeto se mostra bastante promissor, apesar do entrave com relação ao atraso, podendo ser aplicado em várias áreas sem problemas, desde automações para residência até pequenas indústrias. O conhecimento aplicado no projeto é uma porta de entrada para um crescimento mais acelerado da tecnologia USB a situações práticas, pois até o presente momento, era dado uma ênfase maior à comunicação serial, que vem sofrendo declínio devido a expansão do barramento USB.

As possibilidades que surgem com o projeto são ilimitadas, dependendo apenas de ideias inovadoras. Algumas das aplicações sugeridas para dar sequência no projeto seria, um controlador de demanda, muito utilizado hoje para aumentar eficiência da instalação; analisador de qualidade de energia, para verificar distúrbios na energia consumida; supervisório para micro geração, podendo controlar assim a energia produzida e energia consumida diretamente da rede, aproveitando assim a grande expansão da geração distribuída; dentre várias outras aplicações, desde que o projeto possa ser atendido com as atuais limitações.

A ideia após conclusão deste trabalho é prosseguir com os testes experimentais, ou seja, montar a primeira versão do equipamento e refazer os testes para que possa ser verificado se os entraves atuais foram sanados e se novos problemas irão surgir.

Por fim, este trabalho juntamente com o de Germano Bezerra, forma a base para o desenvolvimento de projetos utilizando comunicação USB, já que um utiliza-se da classe CDC e o presente trabalho a classe HID. Ficando assim disponível o conhecimento das duas principais classes utilizadas para comunicação USB.

REFERÊNCIAS

MESSIAS, R. **Curso USB/Serial – Controle de dispositivos Rogercom.** Disponível em: <http://www.rogercom.com/Indice.htm>. Acesso em: 10 de janeiro de 2014.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO. **Universal Serial Bus.** Disponível em: <http://www.gta.ufrj.br/grad/01_1/usb/usb.htm>. Acesso em: 23 de junho de 2014.

COMPAQ; INTEL; MICROSOFT; NEC. **Universal Serial Bus Specification.** Disponível em: <http://mprolab.teipir.gr/vivlio80X86/usb11.pdf>. Acesso em: 24 de junho de 2014.

BEKHIT, A. **USB HID Template for Visual Basic 2005/2008/2010.** Disponível em: <http://helmpcb.com/software/usb-hid-template-for-visual-basic-2005>. Acesso em: 28 de junho de 2014.

ZELENOVSKY, R; MENDONÇA, A. **PC: Um guia prático de Hardware e Interfaceamento**. 4. ed. Rio de Janeiro: MZ Editora, 2006.

MIYADARAI, A. N. **Microcontroladores PIC 18 – Aprenda e Programe em Linguagem C.** 1 ed. São Paulo: Editora Érica, 2009.

PINHO, G. B. M. **Circuito Microcontrolado com comunicação USB para controle de periféricos via interface em Matlab, Brasil.** 2010. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2010. Disponível em: <http://www.dee.ufc.br/anexos/TCCs/2010.2/GERMANO%20BEZERRA%20DE%20MENESES%20PINHO.pdf>. Acesso em: 12 de Outubro de 2014.

FTDI. **Future Technology Devices International Ltd - FT232R USB UART IC.** Disponível em: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>. Acesso em: 12 de Outubro de 2014.

FILHO, J. M. **Modelos de referência e citação com base nas normas da ABNT.** Disponível em: < http://www.sorocaba.unesp.br/Home/Biblioteca/guia-abnt_site.pdf>. Acesso em: 20 de Outubro de 2014.

CASILLO, D. **Automação e Controle Sistemas Supervisórios**. 2011. Aula para o curso Engenharia da Computação, Universidade Rural do Semi-Árido, Mossoró. 2011. Disponível em:<www2.ufersa.edu.br/portal/view/uploads/setores/166/arquivos/Automacao%20e%20Controle%202010_2/Aula%2001%20-%20Automa%C3%A7%C3%A3o%20e%20Controle.pdf>. Acesso em: 4 de Agosto de 2014.

DJ TECHTOOLS. **Figura pág. 19**. Disponível em: < http://www.djtechtools.com/2010/01/25/usb-hub-dj-hi-spee/ >. Acesso em: 10 de Junho de 2014.

ANEXO A – INSTRUÇÕES BIBLIOTECA MCHID.DLL

As informações apresentadas abaixo são de um exemplo de aplicação do MCHID.DLL, disponível no site descrito nas referências (BEKHIT, A,2014):

Visual Basic 2005 HID Functions

How to use:

Before using the code, you need to make sure that the mcHID.dll file is present in either the SYSTEM32 folder of your PC OR in the same directory as your application's executable file. If you plan to distribute your application, the mcHID.dll file must also be included in order for the application to work.

Using the code is very simple. First, all the following variables need to set to their correct values, which will be determined by your USB hardware:

- VendorID
- ProductID
- BufferInSize
- BufferOutSize

The variables can be found at the top of the form code.

If you need to write code that responds to your USB device being plugged or unplugged, simply place that code where shown in the OnPlugged and OnUnplugged events respectively in the main form.

When the USB device sends data to the PC, the OnRead event will be called and the BufferIn array will be populated with the received data. Take note that the received data starts from BufferIn(1) onwards. BufferIn(0) is unused.

If you want to transmit data to the USB device, simply fill the BufferOut array with data, then call the hidWriteEx function to transmit the data to your USB device. Take note that your transmitted data must start from BufferOut(1) and that BufferOut(0) must always be set to 0.

If you want to integrate the code into an existing application, you need to add the mcHIDInterface.vb file to your existing project and copy the code in the template's form into the form in your project that will be doing the USB communication.

Variables:

The following is a brief description of the variables used in the main form that allow USB communication:

- VendorID

Integer value specifying the Vendor ID of the USB device.

- ProductID

Integer value specifying the Product ID of the USB device.

- BufferInSize

Non-zero integer value specifying the size (in bytes) of the data packet that the USB device will send to the PC.

- BufferOutSize

Non-zero integer value specifying the size (in bytes) of the data packet that the PC will send to the USB device.

- BufferIn()

Byte array which contains the data packet received from the USB device. The first byte of the array, BufferIn(0), is unused. Your data will start from BufferIn(1).

- BufferOut()

Byte array which contains the data packet that will be sent to the USB device. The first byte of the array, BufferOut(0), must always be 0. Your data is placed in BufferOut(1) and onwards.

Functions:

The following is a brief description of the important functions that allow USB communication:

- hidWriteEx(ByVal pVendorID As Integer, ByVal pProductID As Integer, ByRef pData As Byte) As Boolean

This function is used to send data to the USB device. It's usage is very simple: the BufferOut array (see above) is filled with the data that needs to be sent (make sure to set BufferOut(0)=0). After that, the function is called as follows:

- hidWriteEx(VendorID, ProductID, BufferOut(0))

VendorID, ProductID and BufferOut are all declared at the top of the form.

Events:

The following is a brief description of the important events in the main form that allow USB communication:

- Form1_Load

When the form is loading, ConnectToHID is called to initialize the USB functions.

This line must be present in order for the USB communication to function properly.

- Form1_FormClosed

When the form is closing, any USB connections are cleaned up and released.

- OnPlugged

This function gets called when your device is plugged into a USB port.

- OnUnplugged

This function gets called when your device has been unplugged from a USB port.

- OnChanged

Not sure what it does, but you don't need to mess with it in order to use USB. Leave it alone and you'll be fine.

- OnRead

This function gets called when data is sent to the PC from your USB device. The data is placed in the BufferIn array declared at the top of the form.

ANEXO B – INSTRUÇÕES DAS BIBLIOTECAS USB PIC

Neste anexo é possível verificar todas as funções disponíveis através das bibliotecas do compilador.

```
////    ****************   USER FUNCTIONS  ************************   ////
////                                                                  ////
//// usb_init() - Initializes the USB stack, the USB peripheral and  ////
////              attaches the unit to the usb bus.  Enables          ////
////              interrupts.                                         ////
////                                                                  ////
//// usb_init_cs() - A smaller usb_init(), does not attach unit      ////
////              to usb bus or enable interrupts.                    ////
////                                                                  ////
//// usb_put_packet() - Sends one packet to the host.                 ////
////                    If you need to send a message that spans     ////
////                    more than one packet then see usb_puts() in  ////
////                    usb.c                                        ////
////                                                                  ////
//// usb_kbhit() - Returns true if OUT endpoint contains data from   ////
////               host.                                              ////
////                                                                  ////
//// usb_rx_packet_size() - Returns the size of packet that was      ////
////                 received.  usb_kbhit() must return TRUE else     ////
////                 this is not valid.  Don't forget in USB there    ////
////                 are 0 len packets!                               ////
////                                                                  ////
//// usb_get_packet() - Gets one packet that from the host.          ////
////                    usb_kbhit() must return true before you call ////
////                    this routine or your data may not be valid.  ////
////                    Once usb_kbhit() returns true you want to    ////
////                    call this as soon as possible to get data    ////
////                    out of the endpoint buffer so the PC can     ////
////                    start sending more data, if needed.          ////
////                    This only receives one packet, if you are    ////
////                    trying to receive a multi-packet message     ////
////                    see usb_gets() in usb.c.                     ////
////                                                                  ////
//// usb_detach() - De-attach USB from the system.                   ////
////                                                                  ////
//// usb_attach() - Attach USB to the system.                        ////
////                                                                  ////
//// usb_attached() - Returns TRUE if the device is attached to a    ////
////                  USB cable.  A macro that looks at the defined  ////
////                  connection sense pin.                          ////
////                                                                  ////
//// usb_task() - Keeps track of connection sense, calling           ////
////              usb_detach() and usb_attach() when needed.          ////
//////////////////////////////////////////////////////////////////////////
```